SIMULATION AND HARDWARE REALISATION OF A TRIPPLE ERROR CORRECTING BCH ENCODER AND DECODER SCHEMES

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by
M. VISHWANADHAM

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
OCTOBER 1977

CERTIFICATE

This is to certify that the work on 'SIMULATION AND HARDWARE REALIZATION OF A TRIPLE ERROR CORRECTING bp BCH ENCODER AND DECODER SCHEMES' has been carried out under my supervision and this has not been submitted elsewhere for a degree.

Vishwanath Sinha

Dr. Viswanath Sinha
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology
Kanpur.

October, 1977.

EE-1977-M-VIS-SIM

ACKNOWLEDGEMENT

I am indebted to Dr. V. Sinha for his guidance, useful discussions and many a helpful suggestion throughout the course of this work. The informal relationship went a long way making the work a pleasure.

I would like to thank my labmates Mr. I.K. Kapur and Mr. M.P. Sastri for making my laboratory work a pleasant and memorable experience.

I would like to record my appreciation for Mr. C.M. Abraha of ACES for his efficient typing of the thesis.

Kanpur,
October, 1977

– M. Vishwanadham

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This thesis deals with an encoding and various decoding schemes for a tripple-error-correcting bp BCH code. The BCH code is chosen for its capability of correcting multiple errors, particularly when the channel corrupts the successive transmitted symbols independently. Two algorithms for BCH decoding due to BerleKamp, Vander Horst and Berger (VDH-B) have been simulated on IBM 7044 for a (31,16) bp BCH code with a minimum distance of 7. It is proved by comparison that VDH-B algorithm is superior to BerleKamp's algorithm. The emphasis has been on finding the usefulness of computerised BCH decoding for use in practical systems. A prototype encoder and decoder scheme has been built and tested for the tripple-error-correcting (31,16) bp BCH code. The decoder uses Chien's cyclic decoding algorithm because of its simple implementation scheme. It has been shown that it is possible to correct all patterns of three or fewer errors. A bit processing rate of 1 Mbit/sec has been realised. For use in practical systems e.g. source encoding such schemes can be readily utilised. The thesis also includes a review of work done on BCH decoding. Relevant simulation programs have been developed.

Chapter 1

INTRODUCTION

1.1 Importance of Coding and Block Codes

The publication of Shannon's existence theorem on coding (1948) and its verification and development of a code by Hamming (1950) started a new era of practical codes for communication purposes. Since then researchers have developed many a code, all of which try to reach Shannon's theorem in one way or other. These efforts have resulted in several good codes which can be implemented. An important class of these under linear block codes are BCH (Bose-Chaudhuri-Hocquenghem) codes.

An (n,k) block code, breaks up the information sequence in to blocks of length k and encodes it in an n-tuple according to certain rules. The extra symbols added are called parity checks and are infact responsible for the error detecting and correcting capabilities of block code.

Until recently use of codes in communications has been limited to areas like deep-space and military communications. In deep-space communication the channel is power constrained and only way to improve SNR and hence reliability is by using coding. In the latter case high reliability was the necessity to use coding. The reasons for not using coding is the cost

of complex encoding and decoding equipment. But with advances in digital technology the additional cost of coding in communications is dwindling fast and use of codes becoming practical.

Today coding is used in computers, in data transmission over telephone-cables, military and space communications. A majority of the codes utilised are block codes. As the improvement in performance of communication systems with coding increases, use of coding will certainly increase.

1.2 BCH Codes

BCH codes are a class of cyclic linear block codes. For any two positive integers 'm' and 't' a BCH code of length $n = 2^m - 1$ exists which corrects all combinations of t or fewer errors and require no more than 'm t' parity-checks. These codes are superior to other block codes in the respect that they require fewer parity checks, for the same codelength and error-correcting capability [10].

All codes of length 15 are optimal and all double error-correcting BCH codes are quasi-perfect and hence optimal also. All BCH codes upto length 1023 are certainly good. But for very large 'n' BCH codes become weak. The lower bound on t/n approaches zero as 'n' becomes large [5,7].

As all digital integrated circuits used in realising encoding and decoding schemes are binary in nature, the binary codes are of prime importance for present day applications. Hence we restrict ourselves in this thesis to binary BCH codes only. Binary BCH codes are particularly useful when the channel noise affects successive bits independently. Further simple implementable decoding procedures have been developed for these codes (Chien, Peterson). These codes also find application in source encoding[3,9].

1.3 Generator of a BCH Code

Definition : All vectors $f(x)$ over GF(2) for which $\alpha, \alpha^2, \alpha^3 \ldots\ldots \alpha^{d-1}$ are roots of $f(x)$ are codevectors of a binary primitive BCH code, where α is a primitive element of $GF(2^m)$. The length of the code is the LCM of the orders of elements $\alpha, \alpha^2 \ldots. \alpha^{d-1}$. d, the design distance is related to error-correcting capability of the code. The fact that a BCH code as defined above will have a minimum distance. 'd' can easily be proved [10].

Letting $d = 2t + 1$

$\{f(x)\}$ is a codevector iff $\alpha, \alpha^2, \alpha^3 \ldots. \alpha^{2t}$ are the roots of $f(x)$. This fact leads us to the generator polynomial $g(x)$. If $m_i(x)$ is the minimum function of α^i then it is also the minimum function of $\alpha^{2^k i}$, $k = 0,1,2 \ldots. m-1$. For example,

the elements α, α^2, α^4, α^8, α^{16} have same minimum function $m_1(x)$ in $GF(2^5)$. Hence an equivalent statement would be $\{f(x)\}$ is a codevector iff α, α^2 ...... α^{2t-1} are the roots of $f(x)$. Thus the generator polynomial of the code is

$$g(x) = LCM[m_1(x), m_3(x), m_{2t-1}(x)] \tag{1}$$

If $m(x)$ denotes a message polynomial then

$$C(x) = m(x) \cdot g(x) \tag{2}$$

is the corresponding BCH codeword. Since $m_i(x)$ cannot have degree more than 'm' the degree of $g(x)$ can atmost be 'mt' and the code has atmost 'mt' parity checks [10].

1.4 Historical Development

BCH codes were developed by Hocquenghem in 1959 and also simultaneously by Bose and Raychaudhuri in 1960 for binary BCH codes which was generalised to non-binary case by Gorenstein and Zierler in 1961. Peterson has developed a correction procedure in 1960. Chien also developed an algorithm for BCH codes in 1964. His algorithm makes use of the cyclic properties of BCH codes and is simple to implement. After Hamming's single correcting codes in 1950, BerleKamp extended them to double error-correcting codes and later came up with a generalised 't' error correcting procedure in 1965. (References are in Peterson). Recent work in this area is due to Vandev Horst and Berger (VDH-B) who published a decoding algorithm for triple error correcting binary primitive BCH codes (1976).

1.5 The present work consists of simulating BerleKamp's and VDH-B algorithms for decoding a (31,16) bp BCH code. The algorithms are compared in performance from simulation results. A prototype encoder and decoder scheme has been implemented in hardware. The decoder makes use of Chien's algorithm.

Chapter 2 discusses BerleKamp and VDH-B algorithms for BCH decoding. Software implementation and some conclusions on their relative performance are also given. Simulation programs are incorporated in appendi~~ces~~x III

Chapter 3 deals with the hardware realisation of (31,16) BCH encoder and decoder. The design equations for the decoder are also given.

In Chapter 4 certain conclusions are drawn on the present work.

Chapter 2

BINARY PRIMITIVE BCH DECODING ALGORITHMS AND SIMULATION

2.1 Introduction

This chapter describes two algorithms for decoding of binary BCH codes. These were simulated on IBM 7044, and their performances compared. The algorithms used are

1. BerleKamp's algorithm
2. Vander Horst and Berger (VDH-B) algorithm

Before explaining these algorithms it would be appropriate to define certain terms pertaining to block codes. For extensive definitions any one of the text should be referred [10].

1. The n-tuple $f = (a_0, a_1, a_2 \ldots, a_{n-1})$ and the polynomial $f(x) = a_0 + a_1X + a_2X^2 + \ldots + a_{n-1}X^{n-1}$ where $a_i \varepsilon (0,1)$ are two identical representations.

2. For an (n,k) block code the ratio (k/n) is defined as the rate of the code. Here 'k' is the no. of information symbols and 'n' the code length.

3. Minimum function of an element β in a $GF(2^m)$ is the monic polynomial of smallest degree with coefficients in the ground field, GF(2) such that

$$m(\beta) = 0$$

4. Every element of an extension field of order 'm' over the ground field has a minimum function of degree m or less.

5. In a GF(2^m) the element α for which $(\alpha^{2^m-1}) = 1$ is known as a primitive element and every non-zero field element can be expressed as a power of α.

2.2 BerleKamp's Algorithm for bp BCH Decoding

Consider a code polynomial $C(X) = \sum_{i=0}^{n-1} C_i x^i$ where $C_i \varepsilon (0,1)$ (1)

During transmission this gets corrupted and let the error sequence be

$$E(x) = \sum_{i=0}^{n-1} E_i x^i \quad , \; E_i \varepsilon (0,1) \tag{2}$$

where if the i^{th} digit is co~~mpu~~rrupted $E_i = 1$, $E_i = 0$. Hence the received polynomial will be

$$R(x) = \sum_{i=0}^{n-1} C_i x^i + \sum_{i=0}^{n-1} E_i x^i \tag{3}$$

As α^i, $i=1,2,..,2t$ are the roots of $g(x)$ and hence of $C(x)$, substituting $x = \alpha^j$ in equation (3), we get

$$R(\alpha^j) = 0 + \sum_{i=0}^{n-1} E_i x^i = \sum_{k=1}^{e} X_k^j = S_j \tag{4}$$

where X_k's are error locations (i.e. at the places $E_k = 1$)

S_j, ($j = 1,2, \ldots\ldots 2t$) are syndromes which are computed by dividing the received polynomial with suitable minimal functions. Thus decoding algorithm will be complete if we find $X_1, X_2, \ldots\ldots X_e$ from the equations

$$\sum_{i=1}^{e} X_i^j = S_j, \; j = 1,2,3 \; \ldots . \; 2t \qquad (5)$$

This is the most difficult part of the algorithm. There are 2t equations and e unknowns ($e \leq t$). Thus the equations (5) will have many solutions each corresponding to a different error pattern in the same coset in the additive group of codewords. The decoder is to be designed to obtain a minimum weight error pattern with the syndrome values. Hence we try to find a solution of (5) with a value of e as small as possible.

An error-locator polynomial is now defined as

$$\sigma(z) = \prod_{i=1}^{e} (1-X_i z) = 1 + \sum_{i=1}^{e} \sigma_j \; z^j \qquad (6)$$

Therefore, the decoder must find σ_j, $1 \leq j \leq t$.

Once $\sigma(z)$ is computed, the reciprocal roots of $\sigma(z)$ will give the error-locations and correction of the received word is accomplished by complementing the erroneous bit, for binary BCH codes. This is achieved by Chiensearch [3].

Chiensearch consists in evaluating $\sigma(X_i^{-1})$ as the digit at i^{th} location X_i leaves the received word buffer and if $\sigma(X_i^{-1}) = 0$ the bit is complemented, otherwise it is passed as it is.

Some mathematical manipulations are necessary to proceed further. Let us define the generating function S(z) as

$$S(z) = \sum_{j=1}^{\infty} S_j z^j = \sum_{j=1}^{\infty} \sum_{i=1}^{e} X_i^j \, z^j = \sum_{i=1}^{e} \frac{X_i z}{1 - X_i z} \qquad (7)$$

Now, multiplying both sides by $\sigma(z)$

$$S(z)\ \sigma(z) = \sum_{i=1}^{e} \frac{X_i z}{(1-X_i z)} \prod_{j=1}^{e} (1-X_j z) = \sum_{i=1}^{e} X_i z \prod_{j \neq i} (1-X_j z) \qquad (8)$$

$$[1 + S(z)]\ \sigma(z) = \sigma(z) + \sum_{i=1}^{e} X_i z \prod_{j \neq i} (1-X_j z) = \omega(z) \qquad (9)$$

It is quite evident that the degree of $\omega(z) \leq e$. Thus

$$[1 + S(z)]\ \sigma(z) = \omega(z) \qquad (10)$$

Syndrome computation on the received word gives us $S_1, S_2, \ldots\ldots S_{2t}$ but $S_{2t+1}, S_{2t+2}, \ldots$ are not available. Therefore, we may write equation (10) as

$$[1 + S(z)]\ \sigma(z) \equiv \omega(z) \bmod z^{2t+1} \qquad (11)$$

Equation (11) was termed as the 'Key Equation' by BerleKamp. Now we can formulate the decoding problem as : Given $S(z)$ we need to compute $\sigma(z)$ and $\omega(z)$ from equation (11). Degree of both $\sigma(z)$ and $\omega(z)$ are $\leq e$, the actual no. of errors occured. BerleKamp developed on algorithm to solve the 'Key Equation'. This is given in the next section.

2.2.1 BerleKamp's algorithm

Initialise $\sigma^{(0)} = 1$, $\tau^{(0)} = 1$, $\omega^{(0)} = 1$, $\gamma^{(0)} = 0$, $D(0) = 0$ and $B(0) = 0$. Proceed recessively as follows :

If S_{k+1} is unknown : Stop

Else define $\Delta_1^{(k)}$ as the coefficient of z^{k+1} in the product $[1 + S(z)]\ \sigma(z)$ and let

$$\begin{aligned} \sigma^{(k+1)} &= \sigma^{(k)} - \Delta_1^{(k)}\ z\ \tau^{(k)} \\ \omega^{(k+1)} &= \omega^{(k)} - \Delta_1^{(k)}\ z\ \gamma^{(k)} \end{aligned} \tag{12}$$

If $\Delta_1^{(K)} = 0$, or, if $D(K) \geq \frac{K+1}{2}$ OR if $\Delta_1^{(K)} \neq 0$ and $D(K) = (K+1)/2$ and $B(K) = 0$

Set

$$\begin{aligned} D(K+1) &= D(K) \\ B(K+1) &= B(K) \\ \tau^{(K+1)} &= z\ \tau^{(K)} \\ \gamma^{(K+1)} &= z\ \gamma^{(K)} \end{aligned} \tag{13}$$

But if $\Delta_1^{(K)} \neq 0$ and either $D(K) < (K+1)/2$ or $D(K) = (K+1)/2$ and $B(K) = 1$.

Set

$$\begin{aligned} D(K+1) &= K + 1 - D(K) \\ B(K+1) &= 1 - B(K) \\ \tau^{(K+1)} &= \sigma^{(K)} / \Delta_1^{(K)} \\ {}^{(K+1)} &= \omega^{(K)} / \Delta_1^{(K)} \end{aligned} \tag{14}$$

The algorithm is continued till $\sigma^{(2t)}$ is found. The explanation of the algorithm and the symbols are available in [1].

2.2.3 Simplifications over the binary field

In case of binary BCH codes $S_1, S_2, \ldots\ldots S_{2t}$ are power symmetric functions of error-locations. Even for $e > t$ the equation

$$S_K = \sum_{i=1}^{e} X_i^K \, , \quad S_{2K} = \left(\sum_{i=1}^{e} X_i^K\right) = S_K^2 \tag{15}$$

holds good. Thus the constraint on the generating function S(z) is that it must satisfy the equation

$$[S(z)]^2 = \sum_{K=1}^{\infty} S_{2K} \, z^{2K} \tag{16}$$

Equation (16) leads to a considerable simplification of BerleKamp's algorithm for binary codes. Also it should be noted that the function $\omega(z)$ which gives the value of the error need not be computed for binary codes. Since we are concerned with binary BCH codes only we give the simplified algorithm in the following section.

Initialise $\sigma^{(0)} = 1 \qquad \tau^{(0)} = 1$

Proceed recursively as follows :

If S_{2K+1} is unknown STOP

Else, define $\Delta_1^{(2K)}$ as the coefficient of z^{2K+1} in the product $[1+S(z)]\ \sigma^{(2K)}$.

$$\text{Compute} \quad \sigma^{(2K+2)} = \sigma^{(2K)} + \Delta_1^{(2K)} \, z \, \tau^{(2K)} \tag{17}$$

$$\tau^{(2K+2)} = \begin{cases} z^2\, \tau^{(2K)} & \text{if } \Delta_1^{(2K)} = 0 \text{ or, if } \deg \sigma^{(2K)} > K \\ \dfrac{z\, \sigma^{(2K)}}{\Delta_1^{(2K)}} & \text{if } \Delta_1^{(2K)} \neq 0 \text{ and } \deg \sigma^{(2K)} \leq K \end{cases} \qquad (18)$$

Continue recursion till $\sigma^{(2t)}$ is obtained.

2.3 VDH-B Algorithm for Tripple Error Correcting bp BCH Codes

A binary n-tuple $x = (x_o, x_1, \ldots. x_{n-1})$, $x_i \in (0,1)$ of weight 'w' can be mapped one-to-one on to a locator polynomial over an extension field $GF(2^m)$, $n = 2^m - 1$ as

$$\sigma(x) = \prod_{i, X_i \neq 0} (X + \alpha^i)$$

where α is a primitive element of $GF(2^m)$

A tripple error correcting bp BCH code can be defined as the null space of a (3xn) matrix H where, its transpose H^T, is given by

$$H^T = \begin{bmatrix} 1 & 1 & 1 \\ \alpha & \alpha^3 & \alpha^5 \\ \alpha^2 & \alpha^6 & \alpha^{10} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \alpha^{n-1} & \alpha^{3(n-1)} & \alpha^{5(n-1)} \end{bmatrix} \qquad (20)$$

The syndromes can be computed by the relation $S = (S_1, S_3, S_5) = RH^T$ where 'R' is the received vector at the decoder input. Now the decoding problem can be viewed as finding a Cosetleader of minimum weight coset C(S) with syndrome S. Equivalently finding a locator polynomial

$$\sigma(X) = \prod_{i=1}^{e} (X + X_i) \tag{21}$$

of minimal degree such that $S_j = \sum_{i=1}^{e} X_i^j$, $j = 1,3,5$ (22)

Here 'e' is the actual number of errors occured in the codeword during transmission. A polynomial such as (21) is known as error-locator polynomial. Thus with every syndrome $S = (S_1, S_3, S_5)$ an error-locator polynomial of minimal degree 'e' is associated. The syndromes and the coefficients of error-locator polynomial satisfy certain relations known as Newton's Identities. These will be given below.

$$\begin{aligned} S_1 &= \sigma_1 \\ S_3 &= \sigma_1 S_1{}^2 + \sigma_2 S_1 + \sigma_3 \\ S_5 &= \sigma_1 S_1{}^4 + \sigma_2 S_3 + \sigma_3 S_1{}^2 + \sigma_4 S_1 + \sigma_5 \end{aligned} \tag{23}$$

After computation of syndromes, equations (23) can be solved and the coefficients of locator polynomial be computed. To do this VDH-B algorithm simplifies Newton's Identities by defining transformed syndrome $t = (T_1, T_3, T_5)$.

Definition : $T_i = S_i + S_1^i$ $(i = 1,3,5)$ (24)

whenever a single error occurs $S_3 = S_1{}^3$ and $S_5 = S_1{}^5$ and S_1 gives the power of α that gives the error location. Hence it is logical to test whether or not $S_3 = S_1{}^3$ and $S_5 = S_1{}^5$ whenever $S_1 \neq 0$. This test is very easy with transformed syndromes. By definition $T_1 = 0$. It remains to check whether $T_3 = T_5 = 0$ or not, when $S_1 \neq 0$. C(t) will be referred to as

a transformed coset : VDH-B algorithm finds a cosetleader for the coset C(s) with syndrome 'S' from the one for C(t) whenever C(s) has a weight $\angle > 1$. In fact, Newton's identities are solved only for transformed syndrome $t = (0, T_3, T_5)$ which are simplified as

$$\begin{aligned} T_1 &= \sigma_1 = 0 \\ T_3 &= \sigma_3 \\ T_5 &= \sigma_2 T_3 + \sigma_5 \end{aligned} \tag{25}$$

2.3.1 Some theorems [11]

A number of theorems are stated below which come handy in developing VDH-B algorithm.

Theorem 1 : If $\sigma(x)$ is any locator polynomial with syndrome S, the

$$\bar{\sigma}(x) = \begin{cases} \sigma(x)/X + S_1 & \text{if } \sigma(S_1) = 0 \\ (X + S_1)\sigma(X) & \text{if } \sigma(S_1) \neq 0 \end{cases} \tag{26}$$

is a locator polynomial with transformed syndrome t. Similarly if $\bar{\sigma}(x)$ is any locator polynomial with the transformed syndrome t, then

$$\sigma(x) = \begin{cases} \bar{\sigma}(X)/(X+S_1), & \text{if } \sigma(S_1) = 0 \\ (X+S_1)\,\bar{\sigma}(X), & \text{if } \bar{\sigma}(S_1) \neq 0 \end{cases} \tag{27}$$

is a locator polynomial with the original syndrome S.

Corrolary : If e and $\bar{e}$ denote the weights of $C(s)$ and $C(t)$ then $|e - \bar{e}| \leq 1$

Theorem 2 : If $\sigma_{2K-1}(X) = \prod_{i=1}^{2K-1} (X + X_i)$, $K \geq 1$ is the locator polynomial of a word in the transformed syndrome 't' and 'L' $\varepsilon\ GF(2^m)$ satisfying $L\ \sigma_{2K-1}(L) = 0$. Then $(X + L)\ \sigma_{2K-1}\ (X + L)$ is also a locator polynomial with syndrome t. Conversely, if $\sigma_{2K}(X)$ is any even degree locator polynomial with syndrome 't' and 'L' is one of its roots, then $\sigma_{2K}(X+L)/X$ also has syndrome 't'.

Theorem 3 : The weight $\bar{e}$ of a transformed coset $C(t)$ is either zero or an odd integer ≥ 3.

Theorem 4 : Let $S = (S_1, S_3, S_5)$ be the syndrome of a coset of weight $e > 1$. Then an error locator polynomial $\sigma(X)$ with syndrome S can be obtained from an error locator polynomial $\bar{\sigma}(X)$ with the transformed syndrome $t = (0, S_3 + S_1^3, S_5 + S_1^5)$ ~~vi**a**~~ via the perscription

$$\sigma(X) = \begin{cases} \bar{\sigma}(X) & \text{if } S_1 = 0 \\ \bar{\sigma}(X)/(X+S_1) & \text{if } S_1 \neq 0,\ e \text{ even} \\ \bar{\sigma}(X+S_1) & \text{if } S_1 \neq 0,\ e \text{ odd.} \end{cases}$$

2.3.2 Error locator polynomials for transformed cosets

Equations (25) give Newton's identities for transformed syndrome 't' $= (0, T_3, T_5)$. We will use those for computing an error-locator polynomial for transformed cosets. Then using

theorem 4 we arrive at an error-locator polynomial with the original syndrome S.

$$T_1 = \sigma_1 = 0, \quad T_3 = \sigma_3, \quad T_5 = \sigma_2 T_3 + \sigma_5 \tag{25}$$

In the general form the error locator polynomial of a transformed coset of weight $\bar{e} = 3$ will be

$$\bar{\sigma}(X) = X^3 + \sigma_2 X + \sigma_3$$

Since $\bar{e} = 3$, $\sigma_5 = 0$ and the use of (25) will give us

$\sigma(X) = X^3 + (T5/T3)X + T3$

The next question is how we get a $\sigma(X)$ from the available $\bar{\sigma}(X)$. This can be done using Theorem 4. From Theorem 4 we arrive at the following :

If $S_1 = 0$ $\sigma(X) = \bar{\sigma}(X)$ and $e = 3$. For $S_1 \neq 0$ we need to consider two cases, i.e. $e = 2$ and $e = 3$.

If $e=2$ then theorem 4 gives us $\sigma(X) = \sigma(X)/(X+S_1) =$

$$X^2 + S_1 X + (T_3/S_1)$$

If $e = 3$ then $\sigma(X) = \sigma(X+S_1) = (X + S_1)^3 + (T_5/T_3)(X+S_1) + T_3$

(30)

If the polynomial does not have three roots in $GF(2^m)$ then $\bar{e} \neq 3$ and from theorem 3, $\bar{e} = 5$, and hence we proceed to the case when $\bar{e} = 5$. Here a test will be necessary for us to verify whether or not $\bar{\sigma}(X)$ has three roots in $GF(2^m)$. One can do it by substituting all non-zero elements of $GF(2^m)$. But a simpler way is available if 'm' is even. We define here

$$\mathrm{tr}(\gamma) = \sum_{i=0}^{m-1} \gamma^{2^i} \qquad \gamma \in GF(2^m) \tag{31}$$

If $1 = \mathrm{tr}[(T_5^3/T_3^5) + 1] = 0$ then we are assured of three roots for $\sigma(X)$ in $GF(2^m)$ [4].

Hence considerable effort is saved by computing the trace before proceeding for a search for roots of $\bar{\sigma}(X)$. If $1 = 1$ we proceed to the case $\bar{e} = 5$. But for odd 'm', $1 = 0$ does not say anything about the roots of $\bar{\sigma}(X)$ in $GF(2^m)$. Then a complete search of $GF(2^m)$ becomes necessary to find the roots of $\bar{\sigma}(X)$. This becomes tedious for large odd 'm'. This fact makes VDH-B algorithm impractical for large odd 'm'.

Let us now consider the case when $\bar{e} = 5$. The general form of error-locator polynomial can be written as

$$\sigma(X) = X^5 + \sigma_2 X^3 + \sigma_3 X^2 + \sigma_4 X + \sigma_5$$

Newtons identities impose

$$\sigma_3 = \sigma_3 \quad \text{and} \quad \sigma_5 = \sigma_2 T_3 + T_5$$

σ_3 and σ_5 are fixed now. We can vary σ_2 and σ_4 arbitrarily over $GF(2^m)$ in a bid to make $\bar{\sigma}(X)$ have fine distinct roots in $GF(2^m)$. A theorem stated below gives us the necessary and sufficients conditions on T_3 and T_5 in order to make $\bar{\sigma}(X)$ has five roots in $GF(2^m)$.

Theorem 5 : If no locator polynomial of degree 3 or less has syndrome $t = (0, T_3, T_5)$, then there is a quintic error locator polynomial $\bar{\sigma}(X)$ with syndrome 't' iff there exist A,B A,B ε GF(2^m) such that

$$\mathrm{tr}\left[\frac{M(A)}{B(B+1)}\right] = \mathrm{tr}\left[\frac{T_3}{A^3}\right] \text{ and } \mathrm{tr}\left[\frac{M(A)}{B(B+1)^3}\right] = \mathrm{tr}\left[\frac{m(A)}{B^3(B+1)}\right] = (\tag{34}$$

where $M(A) = T_3/A^3 + T_3{}^2/A^6 + T_5/A^5$. (35)

When such A,B ε GF(2^m) exist $\bar{\sigma}(X) = (X+A)\ f(X;A,B)\ \ f(X;A,B+1)$ (36

where $f(X;A,B) = X^2 + ABX + A^2\ [T_3/A^3 + B(B+1)\ (\gamma+B^2+1)]$ (37a)

'γ' being either root of the quadratic $X^2+X+M(A)[B(B+1)]^{-1} +$
$[T_3/A^3] + [B(B+1)]^2$ (37b)

It can be seen that the existence of a 'B' that satisfies (34) depends solely on the values of M(A) and $\mathrm{tr}[T_3/A^3]$. Accordingly we define

$\{ C_i = [\eta\ \varepsilon$ GF(2^m) $\ni$ B ε GF(2^m) that satisfies (34) when $M(A) = \eta$ and $\mathrm{tr}(_tT_3/A^3) = i\}$ and $\Psi_i(X) = \prod_{\eta\ \varepsilon\ C_i} (X + \eta)$, $i = 0,1$

By definition of (38), B ε GF(2^m) satisfying (34) for given

A, T_3, T_5 iff $\Psi_i(M(A)) = 0$. (38)

where $i = \mathrm{tr}(T_3/A^3)$, $\Psi_i(X)$ can be expressed as a product of minimal functions of appropriate field elements.

Theorem 6 : X, the minimal polynomial of zero is a factor of $\Psi_0(X)$ but not a factor for $\Psi_1(X)$. That is $0 \varepsilon C_0$ and $0 \notin C_1$.

Theorem 4 says that S_1 is a root of $\bar{\sigma}(X)$ iff $\bar{e} = e + 1$. Thus $\bar{e} = 5$ results in two cases i.e. $e = 4$ and $e = 5$. Find whether or not a 'B' satisfying (34) when $A = S_1$, i.e. seeing whether or not $\Psi_i(M(S_1)) = 0$ where $i = tr(T_3/S_1^3)$. If so, then for any such B

$$\sigma(X) = f(X;S_1,B) \cdot f(X;S_1, B+1), \quad e= 4 \tag{39}$$

where quadratic 'f' is defined by (37). If not then

$$\sigma_1(X) = (X+S_1+A) \cdot f(X+S_1; A,B) \cdot f(X+S_1; A,B+1) \quad e = 5 \tag{40}$$

for any $A,B, \varepsilon GF(2^m)$ that satisfy (34).

2.3.3 The final VDH-B algorithm

From the discussion of error-locator polynomials of transformed cosets an algorithm results for finding $\sigma(X)$ an error-locator polynomial with syndrome S, whenever the maximum coset weight is 5.

VDH-B Algorithm for Decoding of 3-error correcting bp BCH code ($e_{max} = 5$)

Define $\sigma(\cdot)$, $\Psi_i(\cdot)$, $M(\cdot)$ and $f(\cdot)$ respectively by (29), (38), (35), (37).

Step 1 : Compute $T_3 = S_3 + S_1^3$ and $T_5 = S_5 + S_1^5$. If $T_3 \neq 0$ or $T_5 \neq 0$ GO TO Step 2. Else set. $\sigma(X) = X + S_1$ and STOP.

Step 2 : If $tr[(T_5^3/T_3^5) + 1] = 1$, GO TO Step 4. Else. If $\bar{\sigma}(S_1) = 0$ and $tr(T_3/S_1^3) = 0$, Set $\sigma(X) = X^2 + S_1X + (T_3/S_1)$ and STOP.

Step 3 : If $\bar{\sigma}(X)$ has three roots in $GF(2^m)$, Set $\sigma(X) = \bar{\sigma}(X+S_1)$ and STOP.

Step 4 : Let $i = tr(T_3/S_1^3)$. If $\Psi_i[M(S_1)] \neq 0$, GO TO Step 6.

Step 5 : Set $A = S_1$, find B satisfying (34), Set $\sigma(X) = f(X; S_1, B). f(X; S_1, B+1)$ and STOP.

Step 6 : Find A such that $\Psi_i[M(A)] = 0$, find B satisfying (34): Set $\sigma(X) = (X+S_1+A)\, f(X+S_1; A, B).\, f(X+S_1 \cancel{+A}; A, B+1)$ and STOP.

The assumption that maximum coset weight $e_{max} = 5$ assures that A and B with required properties can be found whenever we reach Step 5 or Step 6. Implementation of VDH-B algorithms calls for a foolproof method for finding a 'B' satisfying (34) whenever Step 5 or 6 is reached. In the next section we talk about the software implementation of a (31, 16) bp BCH decoder by means of BerleKamp and VDH-B algorithms.

2.4 Software Implementation

Both the algorithms described earlier in this chapter have been simulated on IBM 7044 for a tripple error correcting (31, 16) bp BCH code. These decoder programs accept as data, an erroneous code word (received word). The programs correct all combinations of three or fewer errors. The received words required as data to decoder simulators have been generated by two programs. The first of these is a (31, 16) BCH encoder simulator program which accepts 16-bit binary sequences as message sequences and generates a BCH code word of length 31. The second program generates error vectors. These are 31-bit binary of tuples with weight $\leqslant 3$. Use of a random number generator available in IBM 7044 subroutine library is made, for getting the binary error sequences. The output of encoder program is added in each position to error vector modulo 2. The resulting received words are fed as data for decoder programs. The time taken for decoder simulator programs for correcting single, double and tri~~p~~ple errors has been computed for both BerleKamp and VDH-B algorithms. It was found that the simulator which uses VDH-B algorithm affects faster correction for the code used. A table of comparison of times taken by both algorithms appears in Table 1. Figures 1 through 4 give the flow charts of the

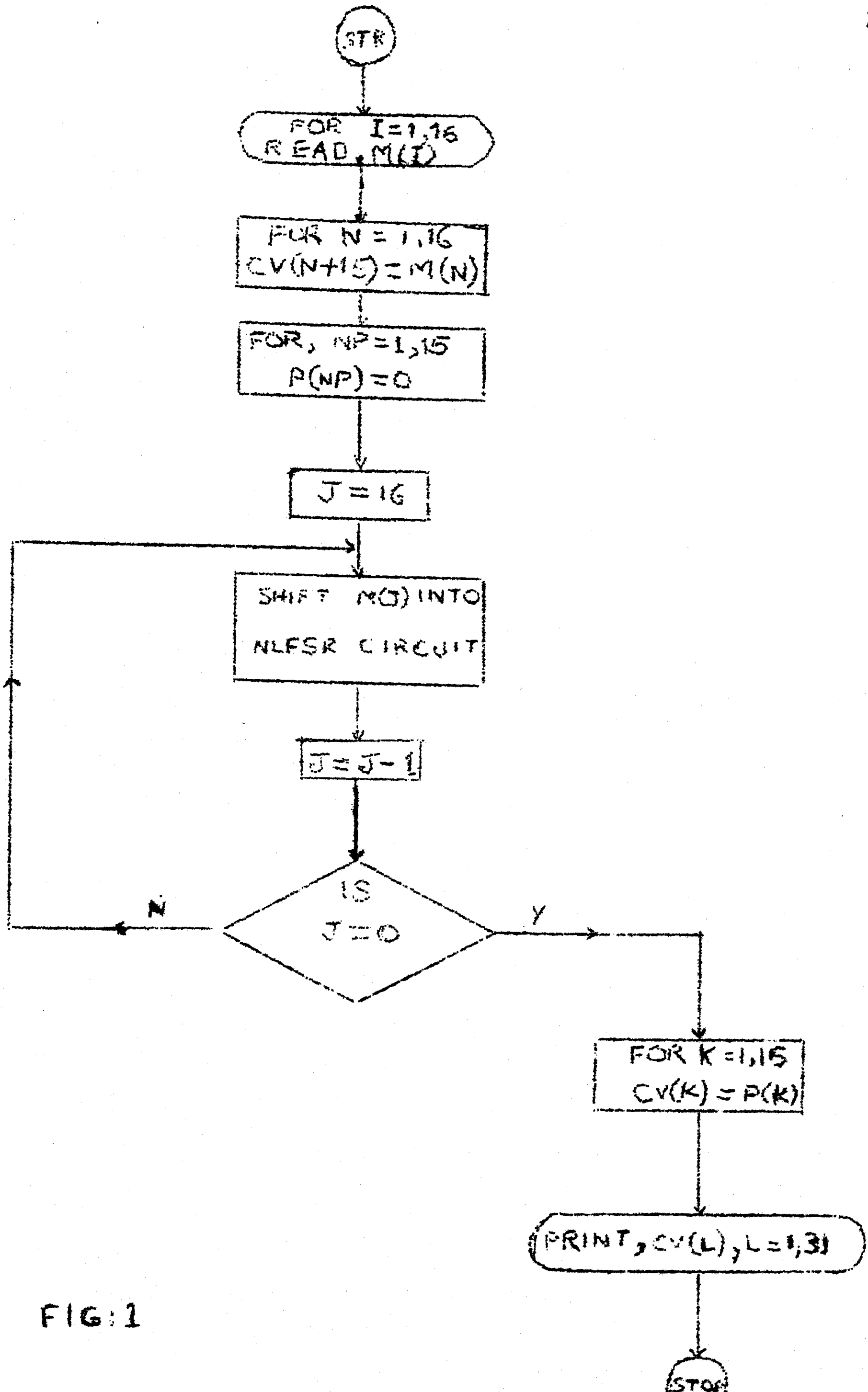


FIG: 1

ENCODER FLOWCHART

ERROR SEQUENCE GENERATOR FLOW CHART

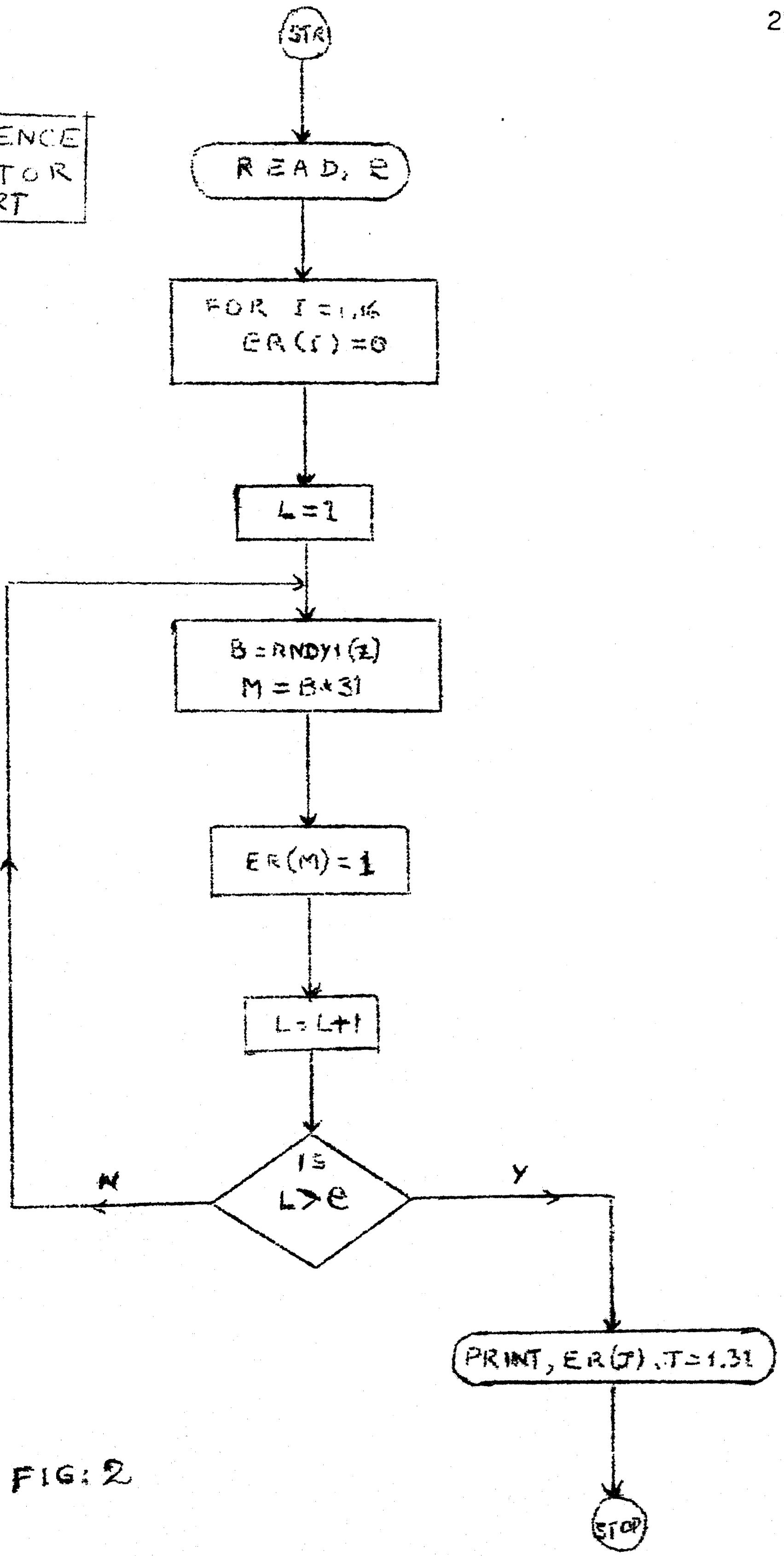


FIG: 2

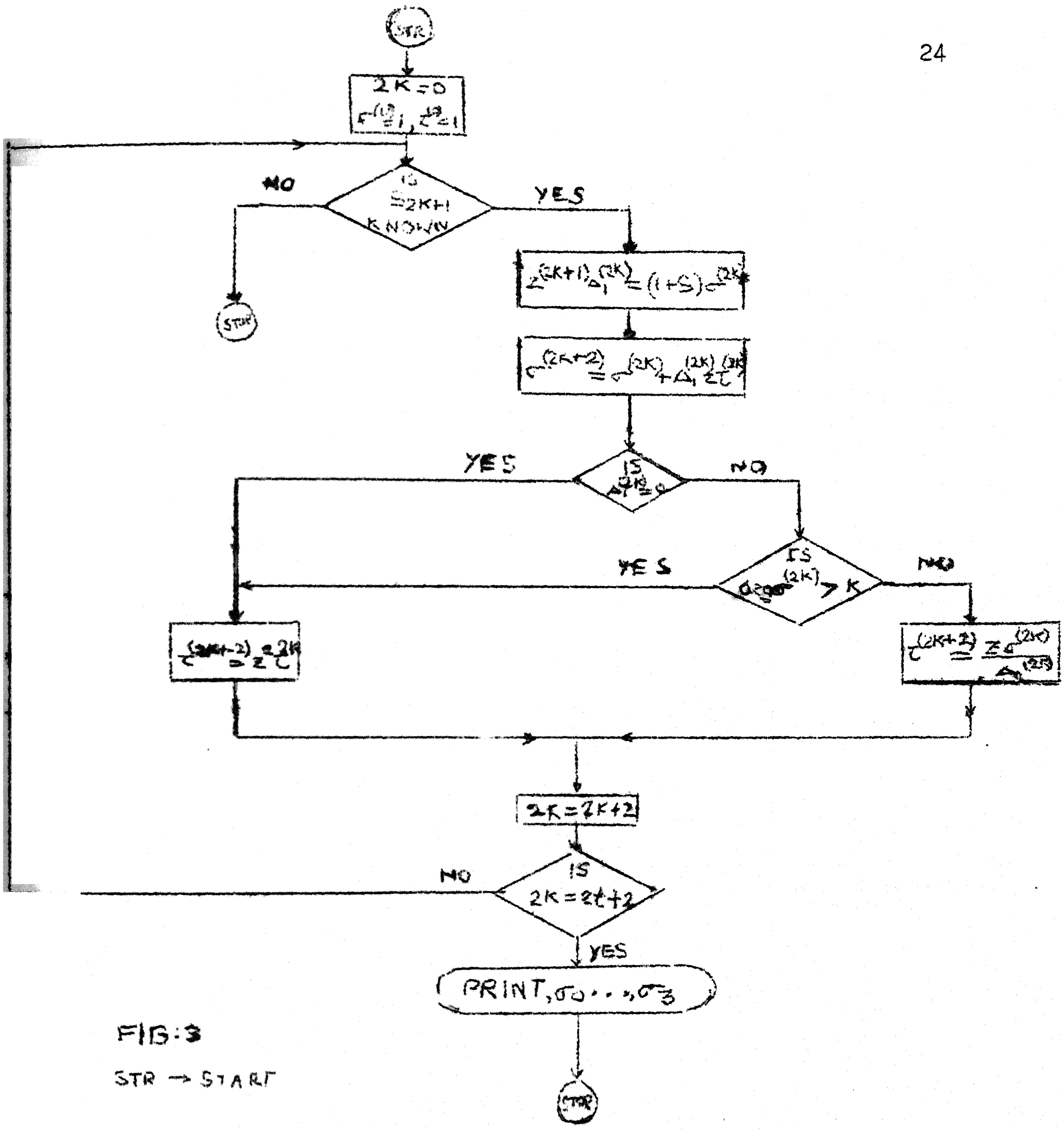


FIG:3

STR → START

VDH-B

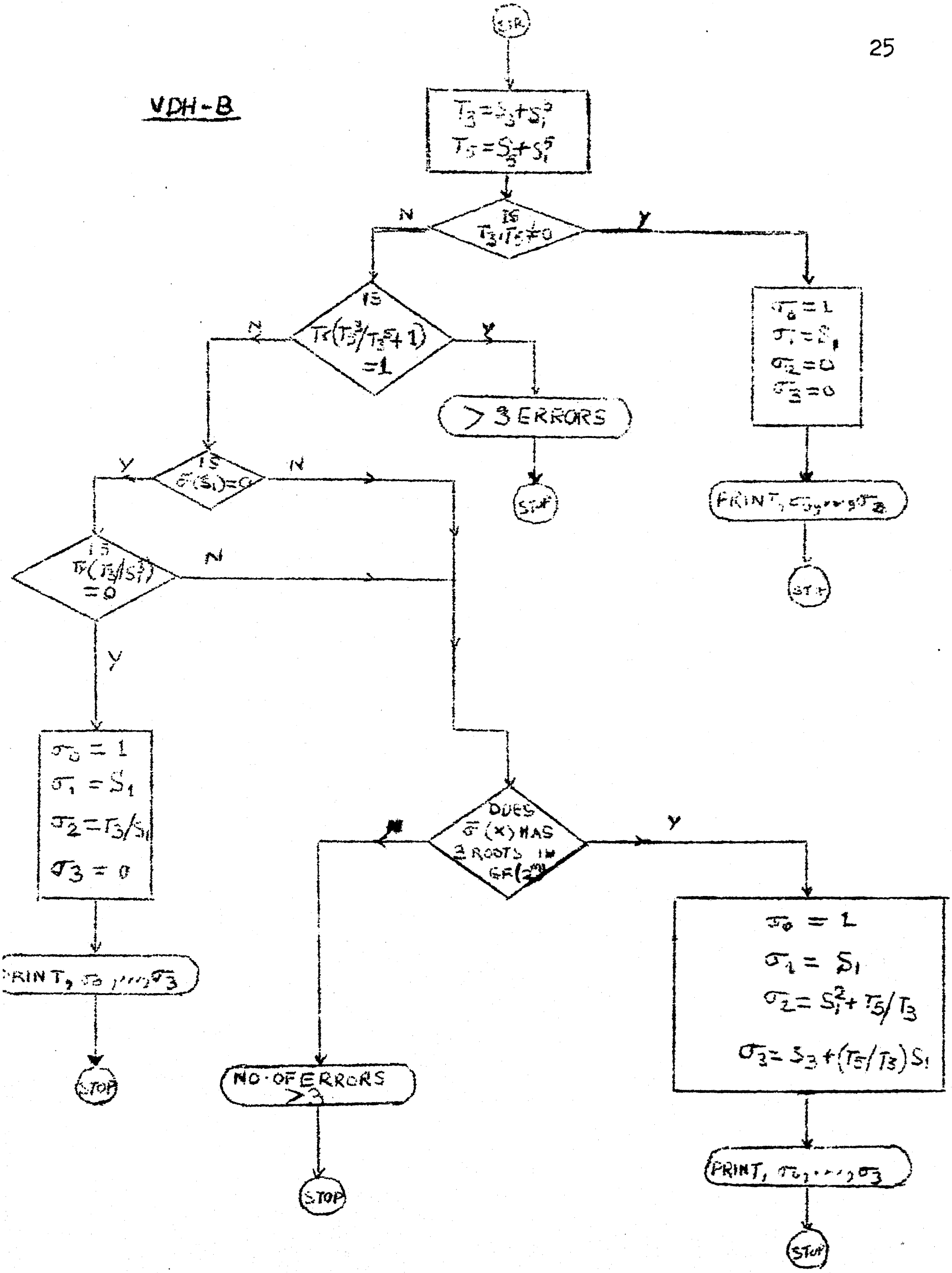

START
$T_3 = S_3 + S_1^3$
$T_5 = S_5 + S_1^5$
IS $T_3, T_5 \neq 0$
N
Y
$\sigma_0 = 1$
$\sigma_1 = S_1$
$\sigma_2 = 0$
$\sigma_3 = 0$
PRINT, $\sigma_0, \ldots, \sigma_3$
STOP
IS $T_5(T_3^3/T_5^5 + 1) = 1$
N
Y
> 3 ERRORS
STOP
IS $\sigma(S_1) = 0$
Y
N
IS $T_5(T_3/S_1^3) = 0$
N
Y
$\sigma_0 = 1$
$\sigma_1 = S_1$
$\sigma_2 = T_3/S_1$
$\sigma_3 = 0$
PRINT, $\sigma_0, \ldots, \sigma_3$
STOP
DOES $\sigma(x)$ HAS 3 ROOTS IN $GF(2^m)$
N
Y
$\sigma_0 = 1$
$\sigma_1 = S_1$
$\sigma_2 = S_1^2 + T_5/T_3$
$\sigma_3 = S_3 + (T_5/T_3)S_1$
PRINT, $\sigma_0, \ldots, \sigma_3$
STOP
NO. OF ERRORS > 3
STOP

FIG. 4

Table 1

For (31,16) bp BCH Code

e	t_c(sec)	
1	BE	0·053
	VDH-B	0·041
2	BE	0·674
	VDH-B	0·506
3	BE	0·763
	VDH-B	0·695

encoder program, random error sequence generator program, BerleKamp's algorithm and VDH-B algorithm respectively. The flow charts are self-explanatory. In case of VDH-B algorithm an attempt to correct 4 and 5 errors in received word is found to result always in an ambiguity for the code used. This is explained in the next section. A table of traces for elements of $GF(2^5)$ appears in appendix I. Simulation programs follow from Appendix III.

2.5 Ambiguities in VDH-B Algorithm

The fact that an attempt to correct a received word with four or five errors using VDH-B algorithm results in ambiguity has been illustrated below by means of two examples. One

example has been considered in each case. The algorithm results in two error-locator-polynomials of minimal degree 4 or 5 in cases where the codeword is corrupted in four and five places respectively. (31,16) BCH code is used.

Example 1 :

Consider a BCH code word C = (0000000000000000000000000000000)

Let the received word be R = (1000000000100000000010000000001)

Computing syndrome $S = (S_1, S_3, S_5) = (\alpha^{12}, \alpha^{20}, \alpha^{2})$ where 'α' is a primitive element of $GF(2^5)$

Computing transformed syndrome $t = (0, T_3, T_5) = (0, \alpha^{29}, \alpha^{8})$

Now we proceed to apply VDH-B algorithm to find a $\sigma(X)$ of minimal degree.

$\Psi_0(X), \Psi_1(X)$ are defined as in Appendix II. $\sigma(X) = X^3 + \alpha^{30}X + \alpha^3$

Step 1 : $T_3 \neq 0$ GO TO Step 2

Step 2 : $Tr[T_5{}^3/T_3{}^5 + 1] = Tr[\alpha^{29}] = 0$, Proceed

$\bar{\sigma}(S_1) = 0$

$Tr[T_3/S_1{}^3] = Tr[\alpha^{24}] = 1$, Proceed .

Step 3 : A search for roots of $\bar{\sigma}(X) = X^3 + \alpha^{10}X + \alpha^8$ in $GF(2^5)$ fails.

Step 4 : $i = 1$, $M(S_1) = (T_3/S_1{}^3) + (T_3{}^2/S_1{}^6) + (T_5/S_1{}^5) = \alpha^{13}$;

$\Psi_1(\alpha^{13}) = 0$ Proceed.

Step 5 : $A = S_1 = \alpha^{12}$

Search for a $B \in GF(2^5)$ that satisfies equation (34) yields $B = \alpha^2$

Computing γ we get $\gamma = \alpha^7$ or α^{22}

With $\gamma = \alpha^7$, $\sigma(X) = (X+\alpha)(X+\alpha^{13})(X+\alpha^{15})(X+\alpha^{23})$ which suggests to correct 2nd, 14th, 16th and 24 positions of the received word, on correction the resulting word $f_1 =$ (1100000000100101000010010000001). It was verified that $f_1 \cdot H^T = 0$, i.e. f_1 is a BCH codeword.

When $\gamma = \alpha^{22}$ we have $\sigma(X) = (X+1)\ (X+\alpha^{10})(X+\alpha^{20})(X+\alpha^{30})$ which tells us to correct first, eleventh, twntwenty-first and thirty-first locations of the received word. The resulting decoded word $f_2 =$ (0000000000000000000000000000000)

$$f_2 \cdot H^T = 0, \text{ i.e. } f_2 \text{ is a BCH code word.}$$

Obviously out of f_1 and f_2 only one of them can be transmitted. Here it is f_2. Hence the ambiguity stated is at hand.

Example 2 :

Consider the BCH code word

$$C = (1000000010111100010011100011011)$$

Let $R = (0000000001111000000111000011010)$

$$S = (S_1, S_3, S_5) = (\alpha^5, 0, \alpha^{11})$$

$$t = (0, \alpha^{15}, \alpha^{24}) \qquad \sigma(X) = X^3 + \alpha^9 + \alpha^{15}$$

Step 1 : $T_3 \neq 0$ GO TO Step 2

Step 2 : $Tr[T_5{}^3/T_3{}^5+1] = Tr(\alpha^{26}) = 1$, GO TO Step 4.

Step 4 : $i = Tr[T_3/S_1{}^3] = Tr[1] = 1$, GO TO Step 6.

Step 6 : Find A and B GF(2^5) which satisfy equation (34).

A search for a results in $A = \alpha^{10}$

With $A = \alpha^{10}$ a search for B results in $B = \alpha^{19}$

$S_1 + A = \alpha^7$

Compute γ

$\gamma = \alpha^{26}$ or α^{28}

With $\gamma = \alpha^{26}$ we get $\sigma(X) = (X+\alpha^3)(X+\alpha^4)(X+\alpha^6)(X+\alpha^7)(X+\alpha^{18})$

i.e. Errors are in fourth, fifth, seventh, eighth and nineteenth positions of the received word. The received word R is decoded as f_1 = (0001101101111000001111000011010)

$f_1 \cdot H^T = 0$, i.e. f_1 is a codeword.

Again $\gamma = \alpha^{28}$ gives us

$$\sigma(X) = (X+1)(X+\alpha^7)(X+\alpha^{16})(X+\alpha^{25})(X+\alpha^{30})$$

Error locations: First, eighth, seventeenth, twenty-fifth and thirty-first

Decoded word f_2 = (1000000101111000100111000111011)

$f_2 \cdot H^T = 0$ i.e. f_2 is a codeword

Once again we are confronted with an ambiguity to decide which one of the codewords f_1 and f_2 has been transmitted.

A reason for this ambiguity can be that , the code used has two cosetleaders for each coset of weight 4 and 5.

This completes our discussion on software implementation of BCH decoding. In the next chapter hardware realisation of a prorotype encoder and decoder for (31,16) bp BCH code are presented with detailed design procedure.

Chapter 3

A PROTOTYPE ENCODER AND DECODER

This chapter describes the design and hardware implimentation of a prototype encoder and decoder scheme for (31, 16) tripple-error-correcting bp BCH code. The encoder employs a 15-bit shift register with nonlinear feedback for parity bit generation. As in any conventional encoding scheme, when all the information bits are fed in to the shift-register (and also simultaneously into the channel), the contents of the shift-register are the parity bits. The codewords generated by this encoder are systematic with information symbols (higher order first) preceeding the parity symbols. The prototype decoder first computes the syndromes and the implements Chien's cyclic decoding algorithm [3] for the correction of all patterns of 3 or fewer errors. A 5 V dc supply ~~is~~ as used, ~~as~~ the scheme has been implemented using TTL IC's. Since the outputs of both encoder and decoder are to be displayed on the CRO, a repetitive pattern is generated by processing the sameword. The important factors for selecting a code are its minimum distance and rate. For larger error-correcting capability very long codes with a large minimum distance are preferable. But such encoder and decoder schemes are complex and costly. For codes of shorter length the rate become unacceptably low. For tripple-error-correction, a (31, 16) BCH code has been selected keeping in view the amount

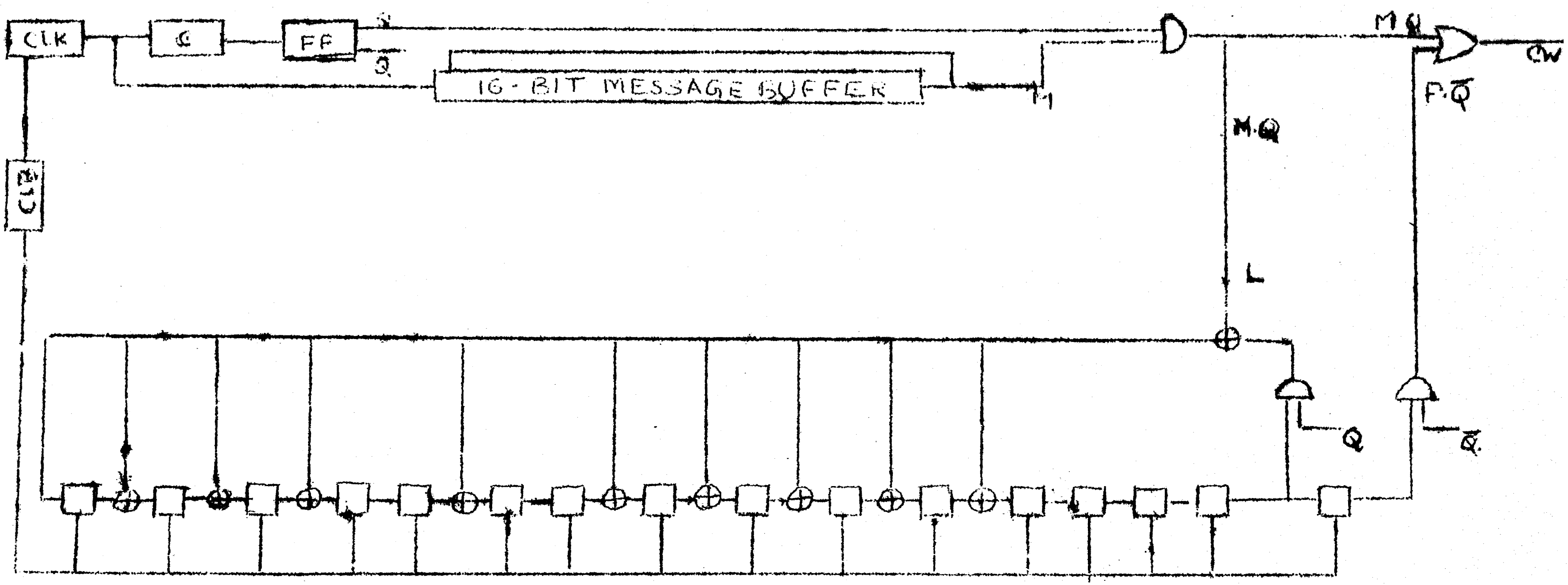
of hardware to be built and also the rate of the code. This code has a reasonably good rate ($k/n = 0.516$). A minimum distance 7 is assured and hence the code corrects all possible patterns of 3 or fewer errors. A correction capability of 3 or less errors is opted, for the code can be used for a noisy channel with $P_e = 0.1382$. In a typical application two standard PCM words (of 8 bits each) can be encoded as a block. Chien's cyclic decoding algorithm [3] is chosen for the decoder mainly for two reasons. One is that the required logic is designed for binary relations and not over $GF(2^n)$. The other reason being automatic correction. By this it means that after syndrome computation error-locator-polynomial is not exclusively computed as in the case of BerleKamp or VDH-B algorithm. This makes the decoder fast operating. The hardware cost is also slightly less than that implementing other algorithms. Hence the choice of the algorithm for hardware realisation.

3.2 Hardware Implementation of (31, 16) bp BCH Encoder

The generator polynomial of tripple-error-correcting (31, 16) bp BCH code is

$$g(X) = m_1(X)\, m_3(X)\, m_5(X) \quad \text{where } m_1(X) = 1 + X^2 + X^5;$$
$$m_3(X) = 1 + X^2 + X^3 + X^4 + X^5;$$
$$m_5(X) = 1 + X + X^2 + X^4 + X^5. \quad \text{Therefore,}$$

(31,16) BCH ENCODER



$$g(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{15}$$

LEGEND

⊕ EX-OR

□ - D-FLIP-FLOP

CLK → CLOCK

CLB → CLOCK BUFFER

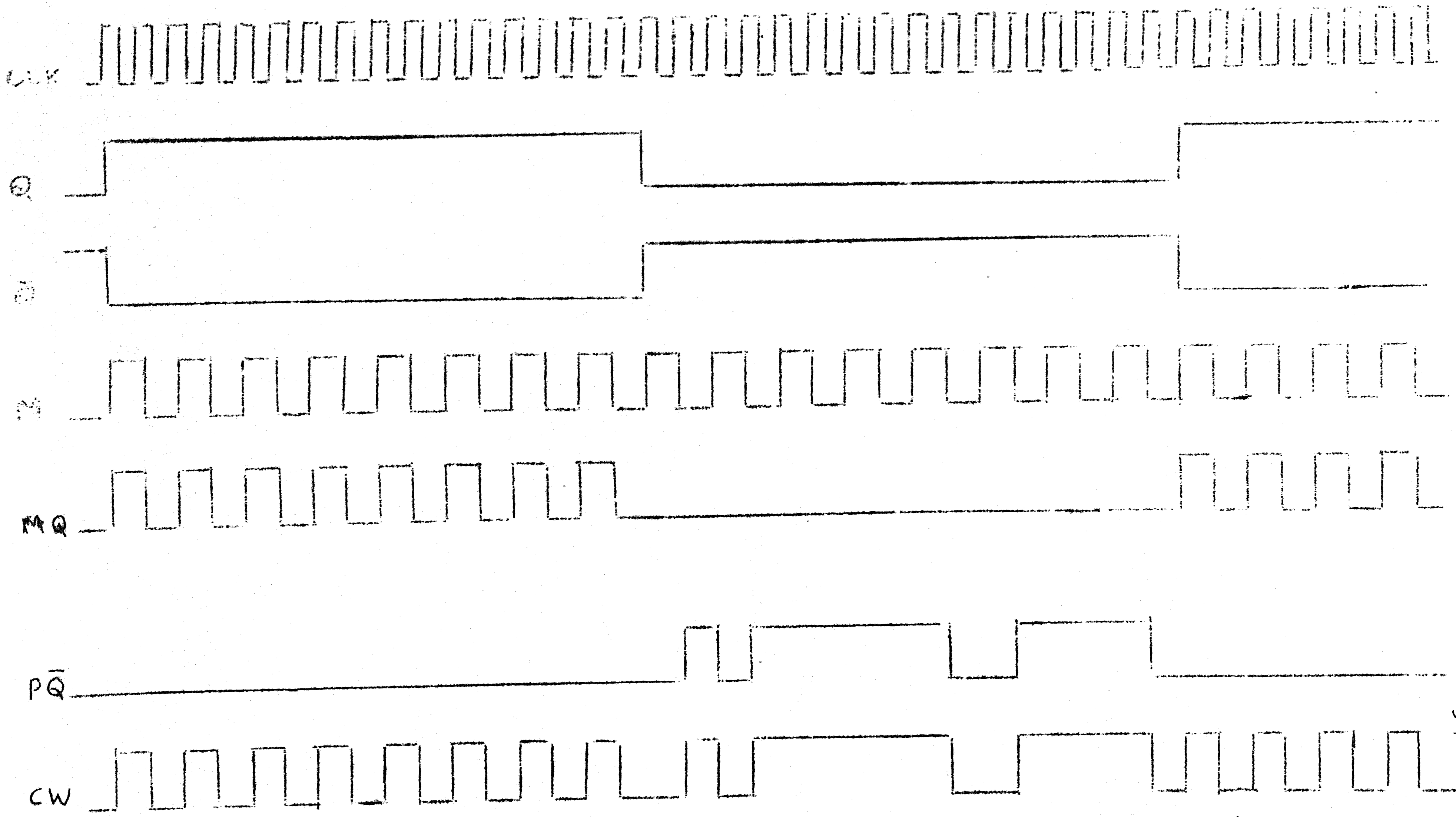
FF – JK FLIP-FLOP

C – 4 bit BINARY

FIG: 5

TIMING FOR THE ENCODER FOR A TYPICAL MESSAGE SEQUENCE

(FIG: 6)



$$g(X) = 1 + X + X^2 + X^3 + X^5 + X^7 + X^8 + X^9 + X^{10} + X^{11} + X^{15} \quad (1)$$

This decides the NLFSR (nonlinear feedback shift-register) circuit needed for check digit generation. The encoder shown in Fig. 5 consists of the following :

1. A 1 MHz system clock
2. A 5-bit binary counter to generate the required control signals
3. A 16-bit buffer register with parallel load capability (for repetitive operation) to hold the information sequence
4. A 15-bit NLFSR (nonlinear feedback shift-register) circuit wired according to g(X), for parity check generation. The buffer register is connected as ringcounter (a circular shift for every clock cycle) so that after 16 clock cycles the contents of the register are the same as in the start. This enables the repetitive encoding of the same information sequence as stated above. The encoder operates in two phases

16 bit information sequence is loaded into the buffer register parallely.

Phase 1 : The contents of buffer register are shifted serially into the channel as well as into the NLFSR circuit, higher order bits shifting first. After 16 such shifts (16 clock cycles) the required parity bits are generated in the NLFSR circuit. Here phase-2 of the encoder operation starts.

Phase 2 : The feedback in the NLFSR circuit is broken at the point 'L' shown in Fig. 5, i.e. the message sequence is inhibited and a zero is placed at point 'L'. Now the check symbols are shifted into the channel with higher order bits entering the channel first. This requires 15 shifts and in the next clock cycle the NLFSR flip-flops are cleared. Encoder has completed encoding one block of information and ready for the next block. Phase 2 ends here. ~~For~~ Both phase 1 and phase 2 required 32 clock cycles in total. Because of 32 shifts the contents of the buffer register are same as in the beginning and again phase 1 and phase 2 are carried out. Thus the output of encoder is a repetitive pattern of period equal to 32 clock cycles. This codeword output can be easily displayed on the CRO. The complete timing diagram for the encoder appears in Fig. 6. The encoder is tested for various information sequences generating valid BCH codewords.

3.3 Considerations for Decoding Algorithm [3]

3.3.1 Use of cyclic property to find error locations

Consider the error-locator-polynomial of degree 't'

$$X^t + \sigma_1 X^{t-1} + \sigma_2 X^{t-2} + \dots + \sigma_t = (X+\beta_1)(X+\beta_2)\dots(X+\beta_t) \quad (2)$$

where β_j's, j = 1,2, ..., t are error locations. Now

$$\sigma_1 = \sum_{j=1}^{t} \beta_j \tag{3}$$

$$\sigma_2 = \sum_{\substack{j,k=1 \\ j \quad k}}^{t} \beta_j, \beta_k \tag{4}$$

$$\sigma_3 = \sum_{i,j,k=1}^{t} \beta_i \beta_j \beta_k$$

and so on

$$\sigma_t = \beta_1 \beta_2 \ldots\ldots \beta_t \tag{5}$$

If β_j $(j = 1,2,\ldots\ldots,t)$ are transformed to $\bar{\beta}_j = \alpha\beta_j$ where α is a primitive element of $GF(2^m)$, one can define a new set of $\tilde{\sigma}_k$'s as functions of $\tilde{\beta}_j$'s in the same way as in (2). σ_k's are homogeneous sums of square free products of roots of order k. Hence the σ_k's and $\tilde{\sigma}_k$'s are related as

$$\tilde{\sigma}_k = \alpha^k \sigma_k \quad (k = 1,2,\ldots\ldots,t) \tag{6}$$

In fact β_j are the roots of the polynomial

$$\Sigma(X) = X^t + \sigma_1 X^{t-1} + \sigma_2 X^{t-2} + \ldots\ldots + \sigma_t \tag{7}$$

After 'τ' transformations

$$\bar{\beta}_j = \alpha^{\tau}\beta_j \qquad j = 1,2, \ldots\ldots, t \tag{8}$$

$$\sigma_k = \alpha^{\tau k}\sigma_k \qquad k = 0,1,2,\ldots\ldots,t \tag{9}$$

Since $n = 2^m-1$, $\alpha^n = \alpha^{(2^m-1)}$ in $GF(2^m)$. Hence the powers never exceed $n = 2^m-1$.

If means are there to find whether or not a particular element of GF(2^n) is a root of the polynomial $\Sigma(x)$, all its roots can be found by counting and by successive transformations. For simple implementation of circuits the unit element is chosen to be the element to be detected. When $\alpha^0 = 1$ is a root of $\Sigma(X)$ we note

$$X^K = 1 \qquad (K = 1, 2, \ldots, t) \tag{10}$$

and $\Sigma(1) = 1 + \sigma_1 + \sigma_2 + \ldots + \sigma_t = 0$

$$\text{or} \quad \sum_{K=1}^{t} \sigma_K = 1 \tag{11}$$

If $\sigma_K = 1$ after $\tau_1, \tau_2, \ldots\ \tau_t$ shifts respectively the roots of $\Sigma(X)$ are $\alpha^{n-\tau_1}, \alpha^{n-\tau_2}, \alpha^{n-\tau_3}, \ldots, \alpha^{n-\tau_t}$. This procedure of finding roots of $\Sigma(X)$ leads to simple implementation. Thus we are able to find error-locations without exclusively solving $\Sigma(X) = 0$.

3.3.2 Decoding algorithm

For applying cyclic decoding algorithm it is necessary only to see whether or not '1' is a root of $\Sigma(X)$, i.e. whether $\sum_{K=1}^{t} \sigma_K = 1$.

Newton's identities (Chapter 2) are stated below for ready reference.

$$S_1 + \sigma_1 = 0 \tag{12}$$

$$S_3 + \sigma_1 S_2 + \sigma_2 S_1 + \sigma_3 = 0 \tag{13}$$

$$S_5 + \sigma_1 S_4 + \sigma_2 S_3 + \sigma_3 S_2 + \sigma_4 S_1 + \sigma_5 = 0 \tag{14}$$

Here $\sigma_1, \sigma_2, \ldots\ldots, \sigma_t$ are unknown and $\sigma_i = 0$ for $i \geq t + 1$. The problem of BCH decoding boils down to solving the equations for σ_k's once the syndromes $S_1, S_3, \ldots\ldots, S_{2t-1}$ are computed. One can proceed to solve (12), (13) and (14) using Matrix methods. In matrix form

$$A\sigma = B \tag{15}$$

where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ S_2 & S & 1 & 0 \\ S_4 & S_3 & S_2 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ S_{2t-1} & S_{2t-3} & S_{2t-4} & S_{t-1} \end{bmatrix} \quad B = \begin{bmatrix} S_1 \\ S_3 \\ S_5 \\ \cdot \\ \cdot \\ \cdot \\ S_{2t-1} \end{bmatrix} \text{ and } \sigma = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \sigma_t \end{bmatrix}$$

If $|A|$ is not zero, we can write

$$\sigma_k = \frac{1}{|A|} \sum_{i=1}^{t} S_{2i-1} A_{i,k} \quad (k = 1,2,3,\ldots,t) \tag{16}$$

where $A_{i,k}$ $(k = 1,2,\ldots.t)$ are the co-factors of the determinant $|A|$. If '1' is a root of (X) then $\sum_{k=1}^{t} \sigma_k = 1$. By substituting for σ_k's in (16) we get

$$\sum_{K=1}^{t} \frac{1}{|A|} \sum_{i=1}^{t} S_{2i-1} A_{i,K} = 1 \tag{17}$$

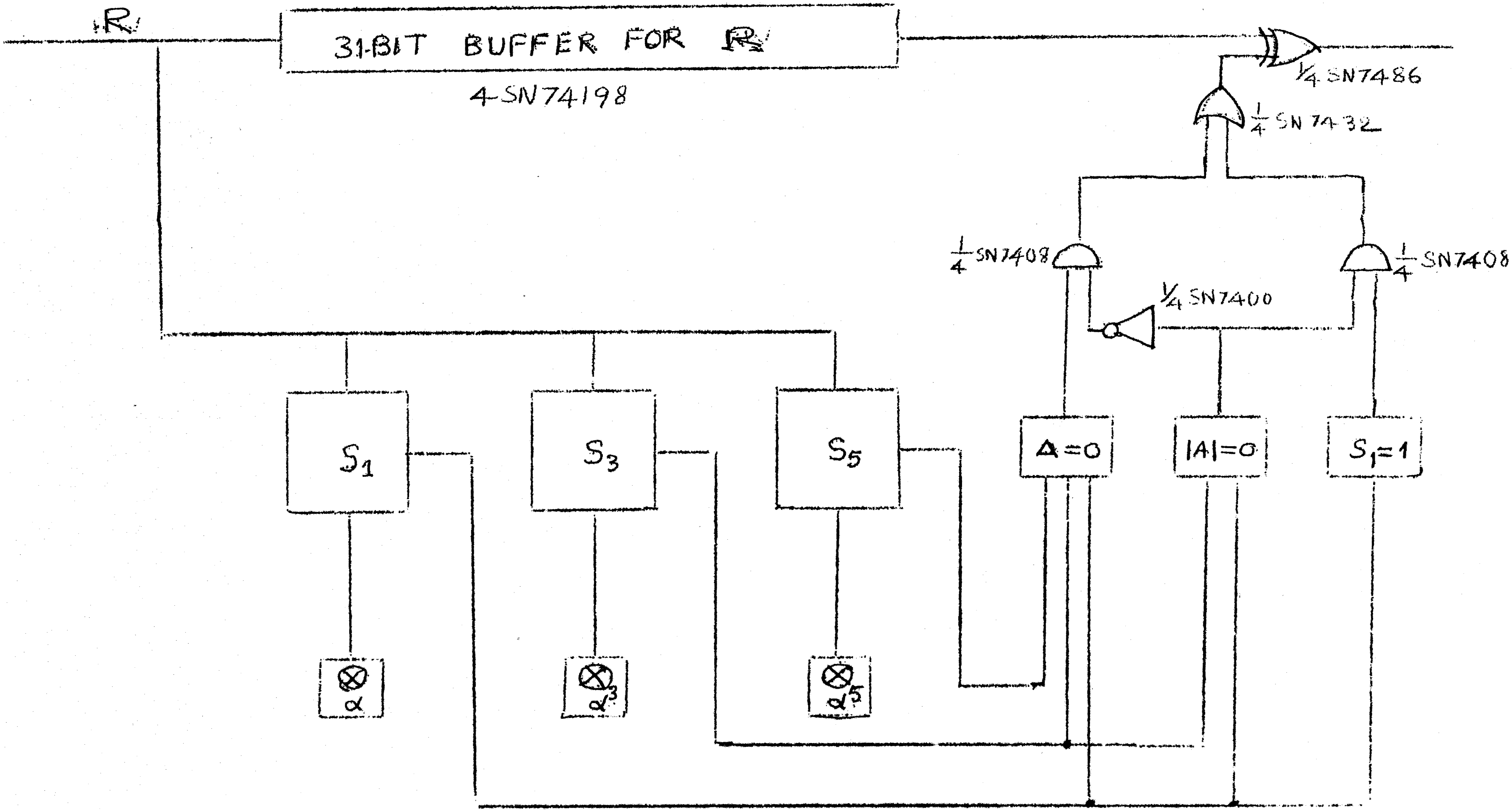
As A is independent of K we write(17) as

$$\sum_{K=1}^{t} \sum_{i=1}^{t} S_{2i-1} A_{i,K} - |A| = 0 \tag{18}$$

Since the characteristic of $GF(2^n)$ is 2, it can be easily shown that (18) is equivalent to setting the determinant to zero, where

$$\Delta = \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ S_1 & 1 & 0 & \cdots & 0 \\ S_3 & S_2 & S_1 & \cdots & 0 \\ S_5 & S_4 & S_3 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ S_{2t-1} & S_{2t-2} & S_{2t-3} & \cdots & S_{t-1} \end{vmatrix} \tag{19}$$

It was shown by Peterson that, $|A| \neq 0$ when the S_i's are power sums of 't' or '(t-1)' distinct roots and $|A| = 0$ when S_i's are power sums of (t-2) or fewer distinct roots [10]. If $|A| = 0$, one can delete the last two equations and end up with (t-2) equations in 't' unknowns. Thus the decoder operates in different modes depending on the size of the largest non-vanishing determinant. The relations such as in (19) are reduced to a set of 'm' binary relations. The circuits are design for these binary relations and not in $GF(2^m)$. This algorithm make a good use of syndrome generator circuits, for error-correction phase.



(FIG: 7)

SYNDROME GENERATOR

CELU (CYCLIC ERROR LOCATION UNIT)

CYCLIC PARALLEL DECODER FOR (31, 16) TEC BCH CODE

$\boxed{\otimes}$ GALOISFIELD MULTIPLIER

3.4 Hardware Implementation of a Prototype (31, 16) bp BCH Decoder

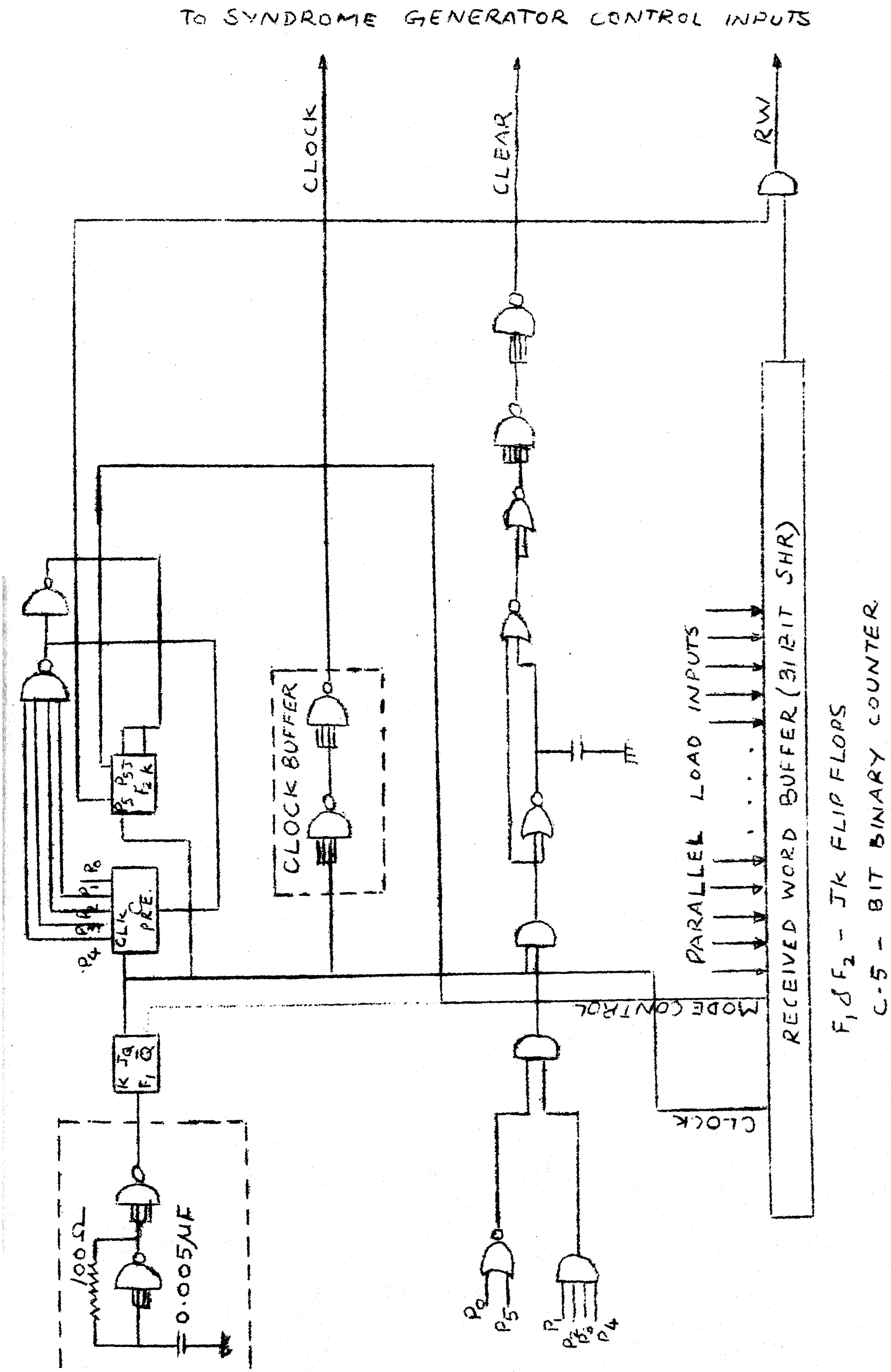
A block diagram of a cyclic decoder for a tri~~p~~ple-error-correcting BCH code is shown in Fig.7. The decoder blocks are

1. A 1 MHz system clock
2. A 31-bit buffer-register to store the received word from the channel
3. Syndrome generator circuits for S_1, S_3, S_5 with associated control logic (Figs. 8 and 9)
4. A cyclic error-location-unit (CELU) which generates the error-pattern accepting the syndrome digits as input

The determinant Δ and $|A|$ of last section for a tri~~p~~ple-error-correcting code are

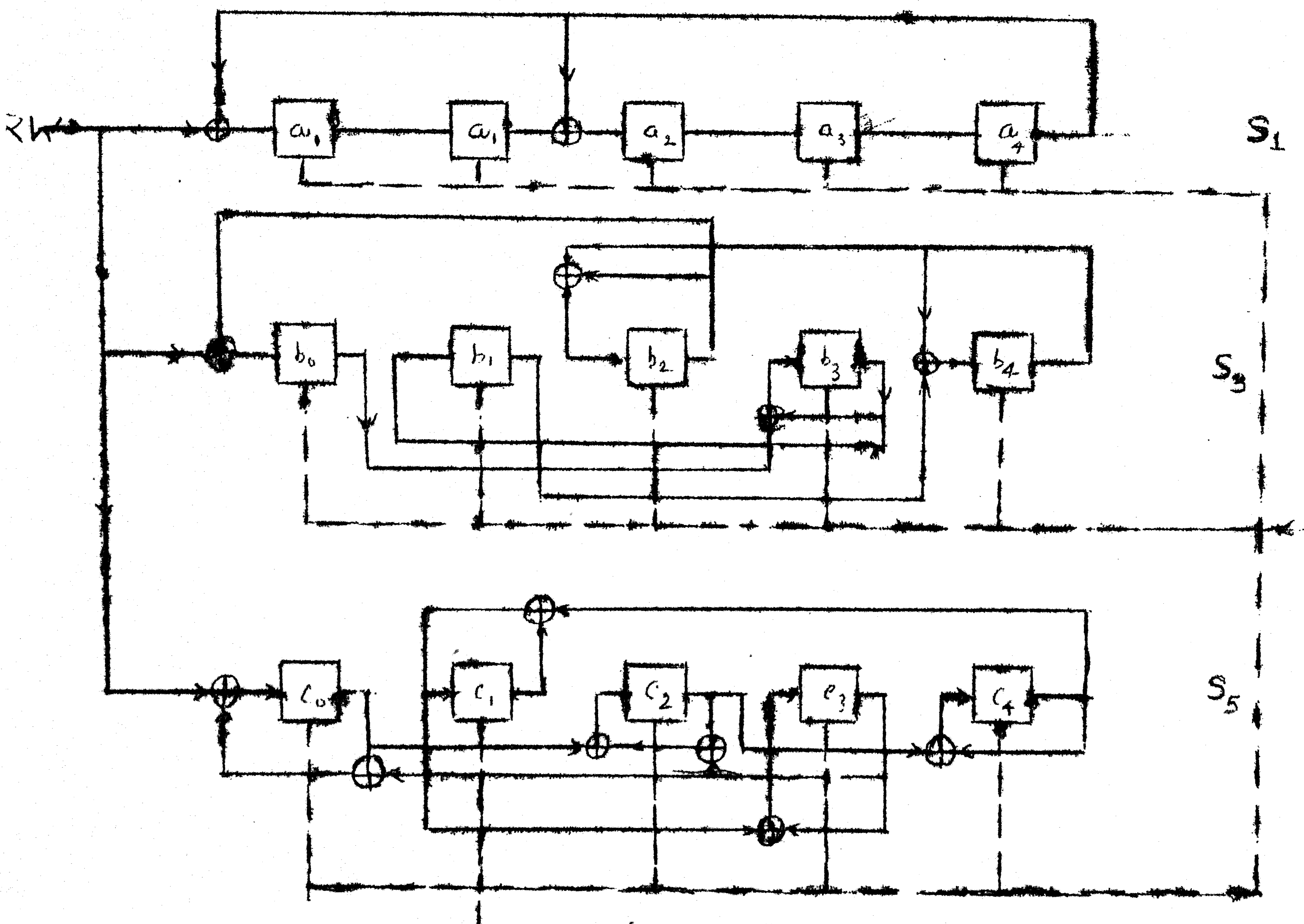
$$\Delta = \begin{vmatrix} 1 & 1 & 1 & 1 \\ S_1 & 1 & 0 & 0 \\ S_3 & S_2 & S_1 & 1 \\ S_5 & S_4 & S_3 & S_2 \end{vmatrix} = (S_1^3 + S_1^4 + S_1^6) + S_3(1+S_1+S_1^2+S_1^3+S_3) + S_5(1+S_1) \qquad (20)$$

$$|A| = \begin{vmatrix} 1 & 0 & 0 \\ S_2 & S_1 & 1 \\ S_4 & S_3 & S_2 \end{vmatrix} = S_1^3 + S_3 \qquad (21)$$



CIRCUIT GENERATES CONTROL SIGNALS FOR SYNDROME GENERATOR

-SYNDROMGENERATORS-



The condition required in error correction is $\Delta = 0$ in case of 3 errors.

$|A| \neq 0$ when 2 or 3 errors occur in the received word

$= 0$ when a single error occurs in the received word

as in case of single error $S_3 = S_1^3$.

All the computations in decoding are in $GF(2^5)$ for the (31,16) code selected. $S_i \varepsilon GF(2^5)$ for $i = 1,3,5$. The elements of $GF(2^5)$ are represented as binary 5-tuples the following representation is used.

$$S_1 = (a_0, a_1, a_2, a_3, a_4)$$
$$S_3 = (b_0, b_1, b_2, b_3, b_4)$$
$$S_5 = (c_0, c_1, c_2, c_3, c_4)$$

where a_i, b_i, c_i, $i = 0,1,2,3,4$ are from $GF(2)$.

3.4.2 Operation of the decoder

To start with a received word of length 31-bits is loaded parallely into the buffer register. The register is connected as a ring counter (last stage output to the input of 1st stage) for the same reason as explained in the case of encoder. The whole decoder can be described as a 62-state sequential machine and works in two phases, each of 31 clock cycles duration. Now the operation of these phases follows :

Phase 1 : Contents of buffer register are circularly shifted. This infact means that the bits from the channel are entering the buffer register as well as three syndrome generator circuits

(fig. 9) wired to divide the incoming received polynomial by $m_1(X)$, $m_3(X)$, $m_5(X)$ respectively. At the end of 31st shift the syndrome generator flip-flops contain the syndrome digits corresponding to S_1, S_3, S_5 respectively. There are 15 syndrome bits with their complements available for use. These outputs drive the CELU circuit. We can describe the state of decoder at the end of 31st shift as follows :

1. The same received word is available in the buffer-register
2. Syndromes $S_1(a_0,a_1,a_2,a_3,a_4)$, $S_3(b_0,b_1,b_2,b_3,b_4)$, $S_5(c_0,c_1,c_2,c_3,c_4)$ with all $\bar{a}_0$, [$\bar{a}_1$ .... $\bar{c}_3, \bar{c}_4$ are available.

At the end of 31st clock cycle the phase 1 is completed. Phase 2 starts with the raising edge of 32nd clockpulse.

Phase 2 : Input to syndrome generator is presented with a zero by using a suitable control logic, at the end of phase 1. As long as zero input is maintained the syndrome generators are simple $GF(2^5)$ counters, the S_1-generator counting the sequence $S1, \alpha S_1, \alpha^2 S_1$ .....$\alpha^{30} S_1$, S_3-generator the sequence $S_3, \alpha^3 S_3$, $\alpha^6 S_3$..... and so on for each clock cycle the contents of S_1, S_3 and S_5 are multiplied by $\alpha, \alpha^3, \alpha^5 \in GF(2^5)$ respectively. This counting operation is shown in Fig. 7 as GF multipliers. This is equivalent to transformation $\bar{\beta}_j = \alpha\beta_j$ explained in the previous section. Such transformation is taking place on

on S_1, S_3 and S_5 for each such transformation 'CELU' circuit produces a '1' or a '0'. During phase-2 also shifting is being carried out in buffer register (but not into syndrome generators). As a bit comes out of buffer 'CELU' circuit produces a '1' if the bit is in error and produces a '0' if the outcoming bit is not corrupted. This process takes place for 31 clock cyclesfrom the completion of phase 1. During this time the output of 'CELU' is a 31-tuple of weight equal to 'e' where 'e' is the actual no. of errors occured in the received word. The output of buffer-register is added to 'CELU' output modulo 2. This results in complementing the buffer output whenever 'CELU' produces a '1', i.e. when the outcoming bit is in error. Thus correction take place in phase 2. At the end of 62nd clock cycle phase 2 is completed.

At this stage once again the same received word is in the buffer-register, phase 1 and phase 2 are carried out alternately, resulting in a repetitive processing of the received word. Testing of decoder is done by feeding different words with different patterns of errors. The decoder does correction whenever the received word is corrupted in 3 or fewer locations. A detailed design and logic functions realised are presented in the following sections.

3.5 Syndrome Generator

The non-zero elements of $GF(2^5)$ are represented by powers of 'α' where 'α' is a root of the irreducible polynomial $1 + X^2 + X^5$ over the ground field. Each power of 'α' is given a binary 5-tuple representation (Appendix IB).

From the definition of syndrome $S_j = R(\alpha^j)$, $j = 1,3,5$. Circuits to evaluate $R(\alpha)$, $R(\alpha^3)$, $R(\alpha^5)$ in $GF(2^5)$ are designed. For each of S_1, S_3, S_5 a nonlinear feed back shift-register, wired according to $m_1(X)$, $m_3(X)$, $m_5(X)$ respectively. Here 'R' is the received word which enters the syndrome circuits with higher order bits first. After 31-shifts the circuits contain the syndrome digits. The syndrome feedback equations are given below. These are GF counters with a polynomial 'R' entering as an input, (Fig. 9).

For $S_1 = (a_0, a_1, a_2, a_3, a_4)$

$$a_0(t+1) = a_4(t) \oplus R(t+1)$$
$$a_1(t+1) = a_0(t)$$
$$a_2(t+1) = a_1(t) \oplus a_4(t)$$
$$a_3(t+1) = a_2(t)$$
$$a_4(t+1) = a_3(t)$$

for $S_5 = (c_0, c_1, c_2, c_3, c_4)$

$$c_0(t+1) = c_0(t) \oplus c_3(t) \oplus R(t+1)$$
$$c_1(t+1) = c_1(t) \oplus c_4(t)$$
$$c_2(t+1) = c_0(t) \oplus c_2(t) \oplus c_3(t)$$
$$c_3(t+1) = c_1(t) \oplus c_3(t) \oplus c_4(t)$$
$$c_4(t+1) = c_2(t) \oplus c_4(t)$$

For $S_3 = (b_0, b_1, b_2, b_3, b_4)$

$$b_0(t+1) = b_2(t) \oplus R(t+1)$$
$$b_1(t+1) = b_3(t)$$
$$b_2(t+1) = b_2(t) \oplus b_4(t)$$
$$b_3(t+1) = b_0(t) \oplus b_3(t)$$
$$b_4(t+1) = b_1(t) \oplus b_4(t)$$

If R=0, for each shift, contents of S_1 are multiplied by 'α', contents of S_3 by 'α^3' and contents of S_5 by 'α^5'. This can be seen by means of an example.

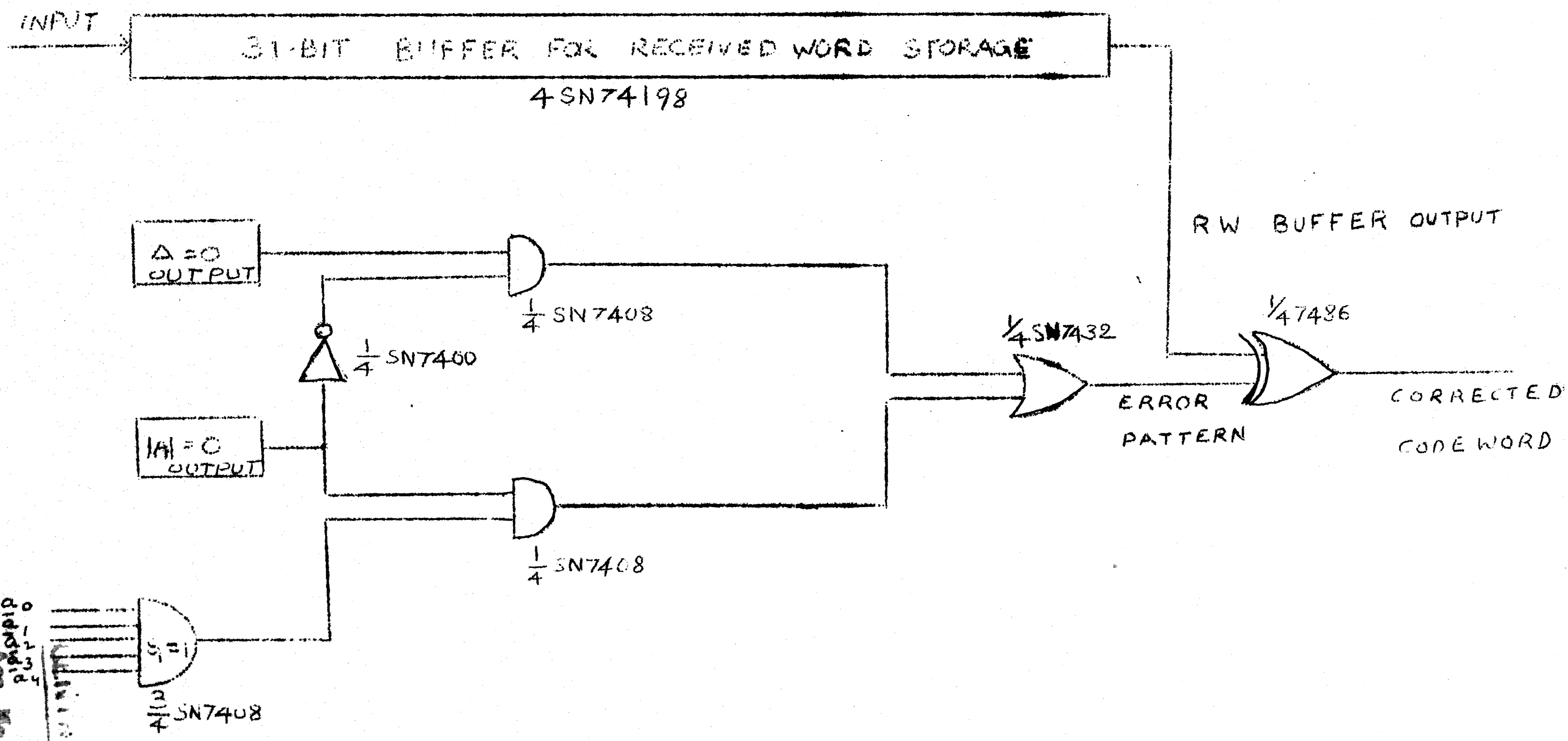
Ex.: $S_3 = 1\ 0\ 1\ 0\ 1 = \alpha^{22}$ and let $R = 0$

Shift the contents of S_3 circularly once. Now using the above equations values of $b_0, \ldots\ldots b_4$ can be computed after the shift operation. If we do this

$$b_0 = 1,\ b_1 = 0,\ b_2 = 0,\ b_3 = 1,\ b_4 = 1$$

Hence contents of 'S_3' after shift operation are $(1\ 0\ 0\ 1\ 1) = \alpha^{25}$. Thus this counter counts in steps of 'α^3'.

The syndrome generator uses D-flip-flops and EX-OR gates (for feedback). The syndrome generators are cleared after correction phase for computing syndromes for next incoming received word (here the same word as processed earlier).

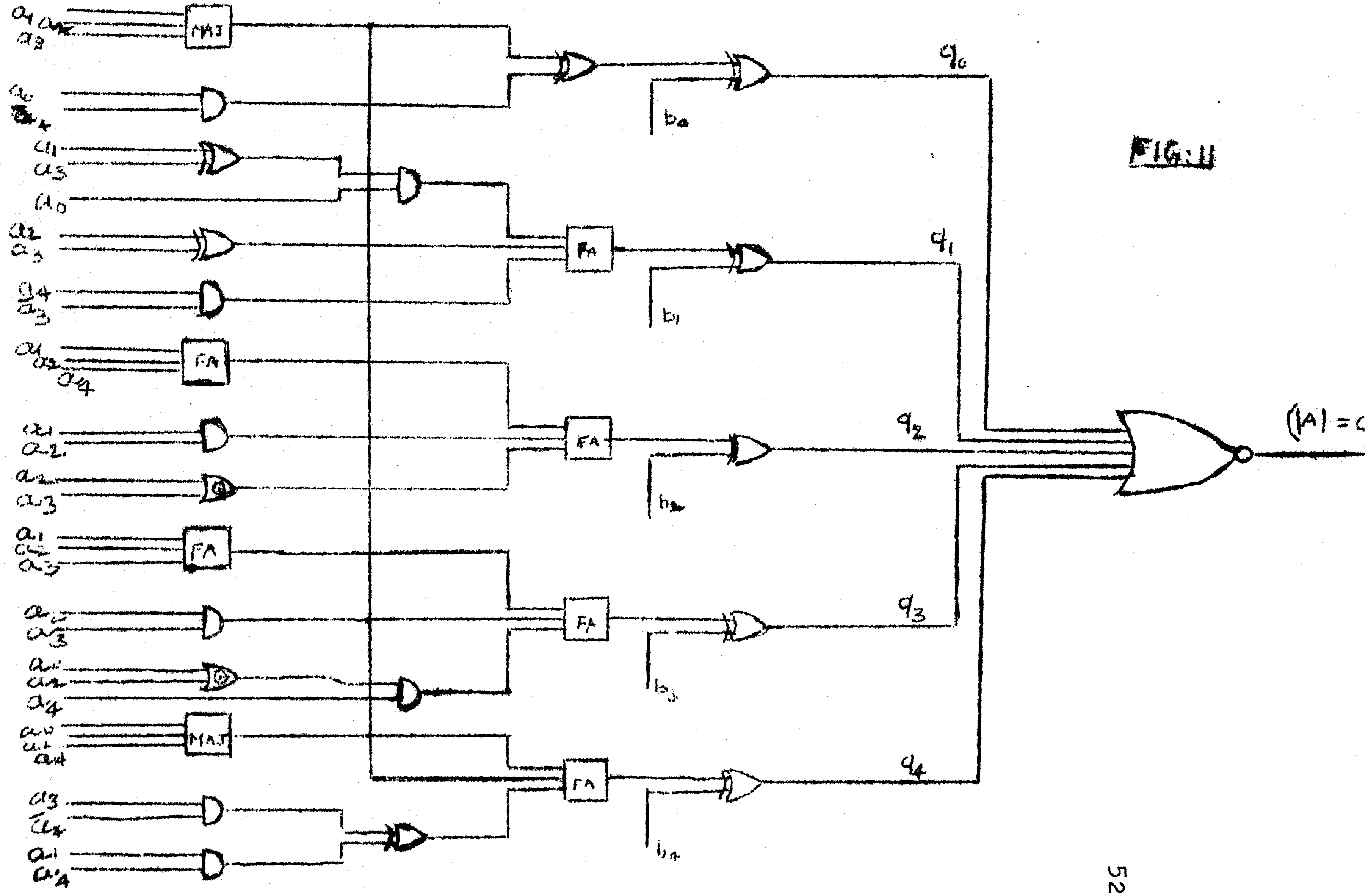


(FIG:10)

CYCLIC ERROR LOCATION UNIT (CELU)

AND ERROR CORRECTION CIRCUIT

FIG: 11



CIRCUIT DETECTS |A| = 0 STATE

3.6 'CELU' Design (Fig. 10)

The cyclic error location unit consists of combinatorial circuits which produce a '1' when $\Delta = 0$, $\mu = 0$ and when $S_1 = 1$ connected as shown in the block diagram of cyclic decoder (Fig. 7).

GF multiplication of any two field elements $(x_0, x_1, x_2, x_3, x_4)$ and $(y_0, y_1, y_2, y_3, y_4)$ is realised by a 10-input combinatorial circuit. The logic required for all these circuits is given below.

3.6.1 Circuit for detecting $\mu = 0$ state (Fig. 11)

$$S_1 = (a_0, a_1, a_2, a_3, a_4)$$
$$S_3 = (b_0, b_1, b_2, b_3, b_4)$$
$$S_1^{\,3} = (P_0, P_1, P_2, P_3, P_4)$$
$$S_1^{\,3} + S_3 = (q_0, q_1, q_2, q_3, q_4)$$

It can be shown that the following hold

LOGIC FOR $P_0, \ldots\ldots P_4$.

$$P_0 = \mathrm{MAJ}(a_1, a_2, a_3) \oplus a_0\bar{a}_4$$
$$P_1 = a_0 \cdot (a_1 \oplus a_3) \oplus a_2 \oplus \bar{a}_4\bar{a}_3 \oplus a_3$$
$$P_2 = a_0 \cdot (a_1 \oplus a_2 \oplus a_4) \oplus a_1 a_2 \oplus a_4 (a_2 \odot a_3)$$
$$P_3 = a_1 \oplus a_2 \oplus a_3 \oplus a_2 a_3 \oplus a_4 (a_0 \odot a_2)$$
$$P_4 = \mathrm{MAJ}(a_1, a_2, a_3) \oplus \mathrm{MAJ}(a_0, a_2, a_4) \oplus (a_3\bar{a}_4 + a_0 a_4)$$

LOGIC FOR $q_0, q_1, \ldots\ldots q_4$

$$q_0 = P_0 \oplus b_0; \quad q_1 = P_1 \oplus b_1; \quad q_2 = P_2 \oplus b_2; \quad q_3 = P_3 \oplus b_3; \quad q_4 = P_4 \oplus b$$

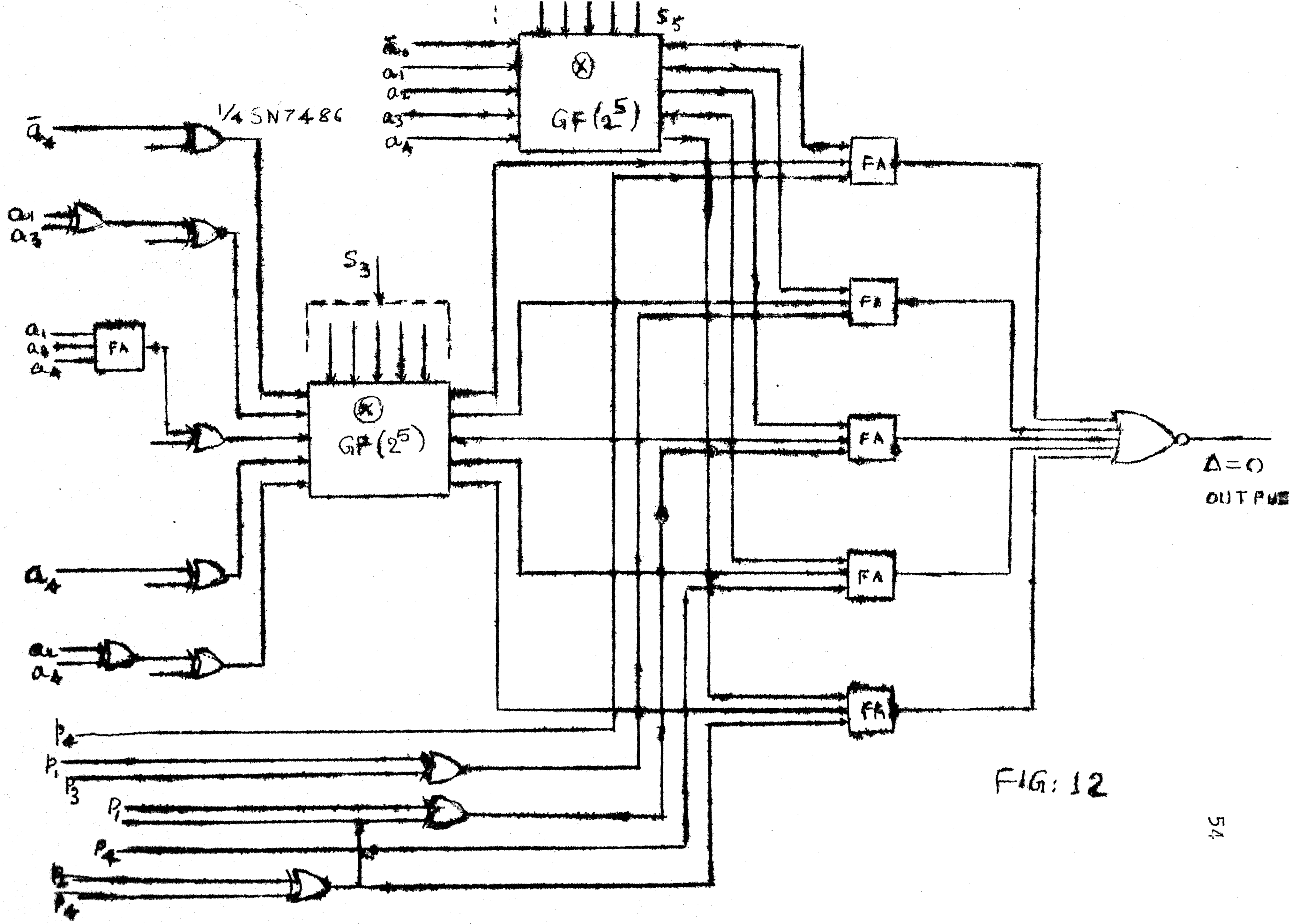


FIG: 12

$\{|A| = 0\}$ OUTPUT IS A LOGICAL '1' WHEN $|A| = 0$ and otherwise a logical '0'. Hence

$$\{|A| = 0\} = q_0, q_1, q_2, q_3, q_4$$

3.6.2 Circuit to detect $\Delta = 0$ (Fig. 12)

$$\Delta = (S_1^3 + S_1^4 + S_1^6) + (1 + S_1 + S_1^2 + S_1^3 + S_3) \cdot S_3 + (1 + S_1) \cdot S_5$$

Let $(1 + S_1 + S_1^2 + S_1^3 + S_3) = (h_0, h_1, h_2, h_3, h_4)$

Logic

$$h_0 = \bar{a}_4 \oplus q_0$$

$$h_1 = a_1 \oplus a_3 \quad q_1$$

$$h_2 = a_1 \oplus a_2 \oplus a_4 \oplus q_2$$

$$h_3 = a_4 \oplus q_3$$

$$h_4 = a_2 \oplus a_4 \oplus q_4$$

The elements $(h_0, h_1, h_2, h_3, h_4)$ and $(b_0, b_1, b_2, b_3, b_4)$ are multiplied using a GF multiplication circuit to realise the second term $(1 + S_1 + S_1^2 + S_1^3 + S_3) \cdot S_3$ in the expansion for . Let

$$(1 + S_1 + S_1^2 + S_1^3 + S_3) \cdot S_3 = (\omega_0, \omega_1, \omega_2, \omega_3, \omega_4)$$

$$1 + S_1 = (\bar{a}_0, a_1, a_2, a_3, a_4)$$

$$S_5 = (c_0, c_1, c_2, c_3, c_4)$$

Another GF multiplication circuit is used to get the term

$(1 + S_1)\ S_5 = (u_0, u_1, u_2, u_3, u_4)$

$(S_1^3 + S_1^4 + S_1^6)$ is generated by a 5 input - 5 output combinatorial network. Let $S_1^3 + S_1^4 + S_1^6 = (v_0, v_1, v_2, v_3, v_4)$

LOGIC

$$v_0 = a_0 \oplus a_2 \oplus a_4 \oplus P_4 \; ; \quad v_2 = a_2 \oplus a_3 \oplus P_1 \oplus P_2 \oplus P_4$$

$$v_1 = a_3 \oplus a_4 \oplus P_1 \oplus P_3 \; ; \quad v_3 = a_2 \oplus a_3 \oplus a_4 \oplus P_4$$

and

$$v_4 = a_2 \oplus P_2 \oplus P_4$$

$$= (\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4)$$

LOGIC

$$\gamma_i = u_i \oplus v_i \oplus \omega_i \quad \text{for } i = 0,1,2,3,4$$

Output should be a logical '1' when = 0 otherwise it is a logic zero

$$(\Delta = 0) = \bar{\gamma}_0 \; \bar{\gamma}_1 \; \bar{\gamma}_2 \; \bar{\gamma}_3 \; \bar{\gamma}_4$$

3.6.3 Circuit to detect $S_1 = 1$ state (Fig. 10)

Here '1' is the unit element of $GF(2^5)$ and has representation (10000)

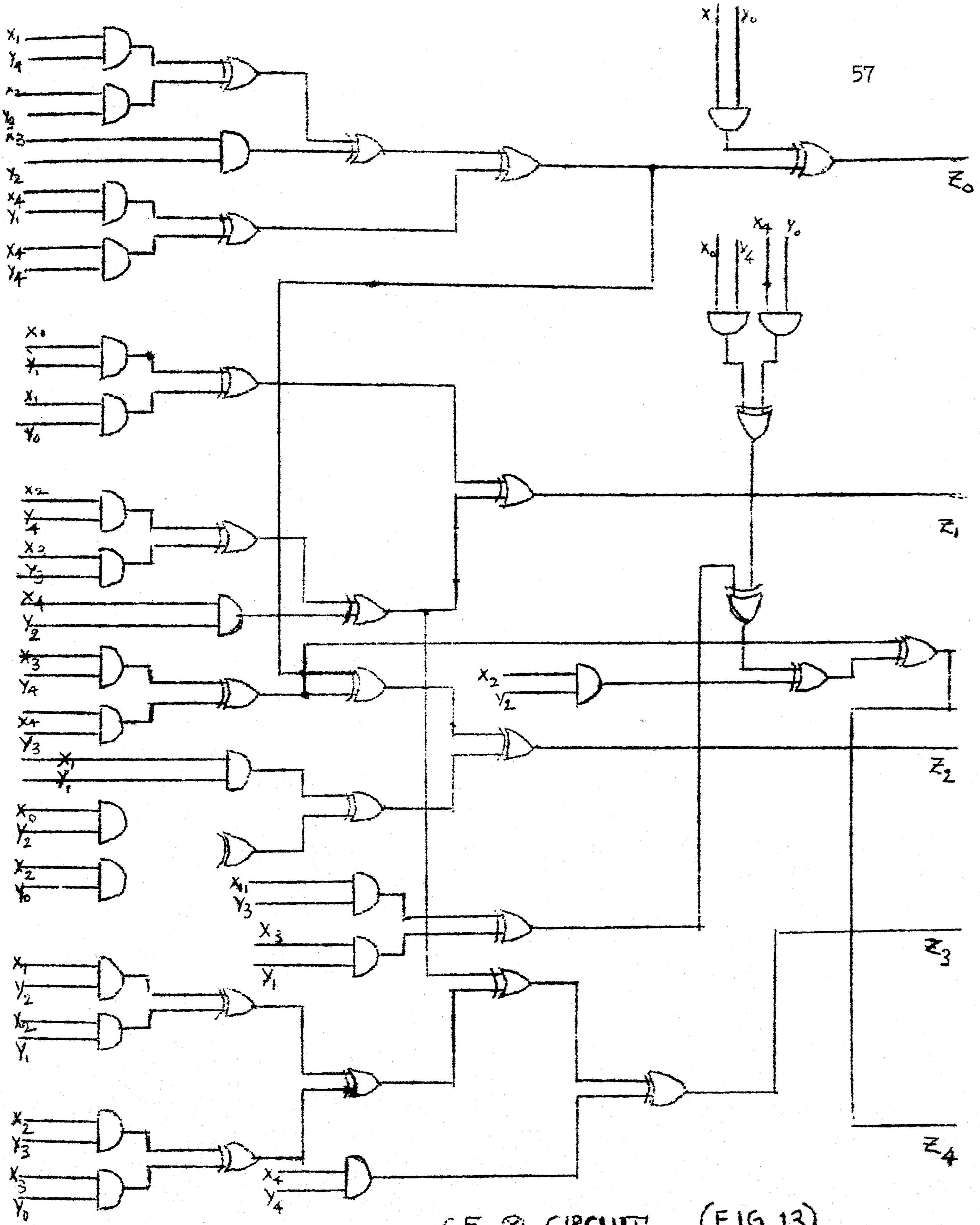
$$(S_1 = 1) \text{ OUTPUT} = a_0 \bar{a}_1 \bar{a}_2 \bar{a}_3 \bar{a}_4$$

3.6.4 GF multiplication circuit (for multiplying A and B ε $GF(2^5)$)

$$A = (x_0, x_1, x_2, x_3, x_4)$$

$$B = (y_0, y_1, y_2, y_3, y_4)$$

Let the product of these elements be another element $C(l_0, l_1, l_2, l_3, l_4)$ ε $GF(2^5)$ then the following combinatorial logic can be arrived at



GF ⊗ CIRCUIT (FIG.13)

A : (X_0 X_1 X_2 X_3 X_4)

B : (Y_0 Y_1 Y_2 Y_3 Y_4) C = AB = (Z_0 Z_1 Z_2 Z_3 Z_4)

$$l_0 = x_0y_0 \oplus x_1y_4 \oplus x_2y_3 \oplus x_3y_2 \oplus x_4y_1 \oplus x_4y_4$$

$$l_1 = x_0y_1 \oplus x_1y_0 \oplus x_2y_4 \oplus x_3y_3 \oplus x_4y_2$$

$$l_2 = x_0y_2 \oplus x_1y_1 \oplus x_1y_4 \oplus x_2y_0 \oplus x_2y_3 \oplus x_3y_2 \oplus x_3y_4 \oplus x_4y_1 \oplus x_4y_3 \oplus x_4y_4$$

$$l_3 = x_0y_3 \oplus x_1y_2 \oplus x_2y_1 \oplus x_2y_4 \oplus x_3y_0 \oplus x_3y_3 \oplus x_4y_2 \oplus x_4y_4$$

$$l_4 = x_0y_4 \oplus x_1y_3 \oplus x_2y_2 \oplus x_3y_1 \oplus x_3y_4 \oplus x_4y_0 \oplus x_4y_3$$

The circuit realised by means of AND and EX-OR gates is shown in Fig. 13.

3.6.5 Summary of hardware realisation

For the encoder 2 PCB's of size (15x11)cm are fabricated one for the buffer register and control logic and the other for parity check circuit. ~~Glass epoxy is the material used for PCB's~~.

For decoder 6 PCB's of size (12.5x16.5)cm are made. The circuits on each PCB is listed below :

Card 1 : $S_1^3 + S_1^4 + S_1^6$ term and also $|A| = 0$

Card 2 and 3 : $GF(2^5)$ Multiplication of any two elements

Card 4 : 'Δ'= 0 circuit and final section of CELU

Card 5 : Received word buffer (31-bit-shift register) and control logic

Card 6 : Syndrome Generator circuits

3.6.6 List of IC's used :

	IC Function	IC No.
1.	Shift registers	SN74198
2.	Binary counters	SN74161, SN7493
3.	D-flip-flops	SN7474
4.	J-K-flip-flops	SN7473
5.	EX-OR Gates	SN7486
6.	Nand Gates	
	2-input	SN7400
	4-input	SN7420
	4-input buffers	SN7440
	8-input	SN7430
	4-input schmitt trigger	SN7413
7.	AND Gates	SN7408
	2-input	
8.	NOR Gates	SN7402
	2-input	
9.	A-O-I Gates	SN7451
10.	Full Adders	SN7480

For more details of the IC's reference is made to any TTL DATA BOOK.

Chapter 4

CONCLUSIONS

This chapter analyses the results reported in the preceeding chapters. Simulation results of two decoding algorithms, due to BerleKamp and VDH-B, showed that VDH-B algorithm is faster and superior to BerleKamp's algorithm. Since most of the digital data processing these days is being done directly on a general purpose computer, the simulation programs developed in fact realises such a system. The simple design procedures in case of Chien's cyclic decoding algorithm has been demonstrated in hardware implementation of the decoder. This algorithm is faster compared to BerleKamp's algorithm as one does not need to compute the error-locator-polynomial exclusively, which require complex logic. BerleKamp's correction procedure needs the computation of the inverse of an element of GF(2^m) which is quite involved, needing sequential logic, where as it is not needed in Chien's algorithm. Cost wise also this algorithm has an edge over BerleKamp's algorithm as most of the circuits use simple logic gates and D-flip-flops. A proto-type encoder and decoder have been built for a (31,16) tri~~p~~ple-error-correcting bp BCH code. The approximate cost of the system is nearly Rs.6000/-. A testing of the system suggests that the prototype can be very easily modified for practical applications like

'source encoding'. In designing the circuitry, emphasis was on using an optimum number of digital IC's and it is felt that the design made conforms to the minimum chip realisation of the system.

References

1. BerleKamp, E.R., 1968b. 'Algebriac Coding Theory', New York: McGraw-Hill Book Company.
2. Bose, R.C. and D.K. Raychaudhuri, 1960a, 'On a class of Error-Correcting-binary Group Codes', Information and Control, 3, pp68-79. (This is the original paper but for our purpose we referred to 'Error Correcting Codes' by Peterson and Weldon).
3. Chien, R.T., 1964b. 'Cyclic Decoding Procedures for Bose-Chaudhuri Chaudhuri Hocquenghem Codes', IEEE Trans. IT-10, pp 357-363.
4. Chien, R.T., B.D. Cunningham and I.B. Oldham, 1969, 'Hybrid methods for finding roots of a polynomial - with application to BCH Decoding', IEEE Trans., IT-15, No.2, pp 329-334.
5. Gorenstein, D., W.W. Peterson and N. Zierler, 1960, 'Two-Error Correcting Bose-Chaudhuri codes are quasi-perfect', Information and Control 3, pp 291-294.
6. Lin, S., 1970, 'Introduction to Error-Correcting Codes', Englewood Cliffs, N.J. Prentice Hall Inc.
7. Lin, S. and E.J. Weldon Jr. 1967b, 'Long BCH Codes are bad', Information and Control, 11 p 445.
8. Massey, J.L., 1969. 'Shift Register Synthesis and BCH Decoding', IEEE Trans., IT-15, No.1, pp 122-127.
9. Peterson, W.W. 1960a, 'Encoding and Error-Correction procedures for the Bose-Chaudhuri Codes', IRE Trans., IT-6, pp 459-470.

10. Peterson W.W. and E.J. Weldon, Jr., 1972, Cambridge, Mass : The M.I.T. Press. "Error Correcting Codes"

11. Vander Horst and Toby Berger, 'Complete Decoding of Tripple-Error-Correcting Binary BCH Codes', IEEE Trans. IT-22, pp 138-147.

Appendix I

GF(2^5) ELEMENT	BINARY REPRESENTATION	GF(2) TRACE
1	1 0 0 0 0	1
α	0 1 0 0 0	0
α^2	0 0 1 0 0	0
α^3	0 0 0 1 0	1
α^4	0 0 0 0 1	0
α^5	1 0 1 0 0	1
α^6	0 1 0 1 0	1
α^7	0 0 1 0 1	0
α^8	1 0 1 1 0	0
α^9	0 1 0 1 1	1
α^{10}	1 0 0 0 1	1
α^{11}	1 1 1 0 0	1
α^{12}	0 1 1 1 0	1
α^{13}	0 0 1 1 1	1
α^{14}	1 0 1 1 1	0
α^{15}	1 1 1 1 1	0
α^{16}	1 1 0 1 1	0
α^{17}	1 1 0 0 1	1
α^{18}	1 1 0 0 0	1
α^{19}	0 1 1 0 0	0
α^{20}	0 0 1 1 0	1

contd.....

Appendix I (contd....)

GF(2^5) ELEMENT	BINARY REPRESENTATION	GF(2) TRACE
α^{21}	0 0 0 1 1	1
α^{22}	1 0 1 0 1	1
α^{23}	1 1 1 1 0	0
α^{24}	0 1 1 1 1	1
α^{25}	1 0 0 1 1	0
α^{26}	1 1 1 0 1	1
α^{27}	1 1 0 1 0	0
α^{28}	0 1 1 0 1	0
α^{29}	1 0 0 1 0	0
α^{30}	0 1 0 0 1	0

Appendix II

$\Psi_0(X)$, $\Psi_1(X)$ AS PRODUCTS OF MINIMAL POLYNOMIALS OF $GF(2^m)$

$m_i(X)$ IS MINIMAL POLYNOMIAL OF α^i

TABLE 2

m	$\Psi_0(X)$	$\Psi_1(X)$
4	$X\, m_0(X)m_5(X)$	$m_1(X)$
5	$X\, m_5(X)m_{11}(X)m_{15}(X)$	$m_1(X)m_5(X)m_7(X)m_{11}(X)$
6	$(X^{64}+X)/m_{13}(X)m_{31}(X)$	$(X^{63}+1)/m_1(X)m_5(X)m_9(X)$
7	$X^{31}+1$	$X^{31}+1$
$8 \leq m \leq 12$	$X^{n+1}+X,\ n = 2^m - 1$	$X^n + 1,\ n = 2^m - 1$

```
SOME IMPORTANT DEFINITIONS.
CHARACTERISTIC OF GALOIS FIELD USED --- 2
M --- ORDER OF THE GALOIS FIELD USED IN COMPUTATIONS.
N --- LENGTH OF THE CODEWORD.
K --- NUMBER OF INFORMATION SYMBOLS(BITS).
(N-K) --- NUMBER OF PARITY CHECK SYMBOS(BITS).
N= (2**M)-1
M=5,K=16,N=31,(N-K)=15 FOR THE BINARY PRIMITIVE BCH CODE USED.
P(I) --- D- FLIP-FLOP CARRYING I'TH PARITY BIT.
R --- A BUFFER REGISTER.
M --- INFORMATION SEQUENCE.
CV --- BCH CODEWORD PRODUCED BY THE ENCODER.
-----------------------------------------------------------------------
S I M U L A T I O N  P R O G R A M  FOR (31,16) BCH  E N C O D E R
-----------------------------------------------------------------------
INTEGER   P(15),R(15),M(16),CV(31)
DATA   MN/10/
DO 7 IR = 1,MN
THE MESSAGE BLOCK IS READ.
READ10,(M(IT),IT=1,16)
PARITY BIT GENERATOR INITIALISED TO ALL ZEROES.
DO 6 IX = 1,15
P(IX) = 0
THE MESSAGE SEQUENCE IS SHIFTED INTO PARITY GENERATOR CIRCUIT WITH
HIGHER ORDER BITS ENTERING FIRST,AND ALSO INTO THE CHANNEL
SIMULTANEOULY. AFTER '16' SUCH SHIFTS THE CONTENTS OF
PARITY GENERATOR ARE THE REQUIRED PARITY BITS.
DO 1 J = 1,16
IQ = J+15
CV(IQ) = M(J)
```

```
  2   R(1) = MOD2(M(L-1)+P(15))
      R(2) = MOD2(R(1)+P(1))
      R(3) = MOD2(R(1)+P(2))
      R(4) = MOD2(R(1)+P(3))
      R(5) = P(4)
      R(6) = MOD2(R(1)+P(5))
      R(7) = P(6)
      R(8) = MOD2(R(1)+P(7))
      R(9) = MOD2(R(1)+P(8))
      R(10) = MOD2(R(1)+P(9))
      R(11) = MOD2(R(1)+P(10))
      R(12) = MOD2(R(1)+P(11))
      R(13) = P(12)
      R(14) = P(13)
      R(15) = P(14)
      DO  3  K = 1,15
  3   P(K) = R(K)
      L = L-1
      IF(L.GT.1)  GO  TO  2
C     PARITY BIT GENERATION  IS  COMPLETE.
C     PARITY  BITS  ARE  SHIFTED  INTO THE CHANNEL  WITH  HIGHER  ORDER
C     BITS  ENTERING  FIRST.
      DO  5  N = 1,15
  5   CV(N) = P(N)
C     THE MESSAGE SEQUENCE  AND  THE  CORRESPONDING CODEWORD ARE PRINTED.
      PRINT30,(M(IT),IT=1,16),(CV(JL),JL=1,31)
C     PARITY GENERATOR IS CLEARED FOR ENCODING THE NEXT MESSAGE BLOCK.
C     ENCODING IS COMPLETE AND THE ENCODER IS READY TO ENCODE NEXT MESSAGE BLOCK
  7   CONTINUE
 10   FORMAT(16I1)
 30   FORMAT(1H0,10X,16I2,10X,31I2)
      STOP
```

```
THIS  PROGRAM  CORRESPONDS  TO  THE  FIGURE  NO. '2'  OF  THE  THESIS
THE   PROGRAM  GENERATES  RANDOM  BINARY  SEQUENCES  OF  WEIGHT  'M'
AND  LENGTH  31.  R(31)  IS  THE  ERROR  VECTOR  GENERATED.
INTEGER    R(31)
DO  1  K = 1,31
R(K) = 0
DO  2  I = 1,M
X = RANDY1(Z)
Y = X*31.
J = Y
IF(J.EQ.0)  J = J+31
R(J) = 1
PRINT8,(R(L),L=1,31)
FORMAT(1H0,25X,31I2)
STOP
END
TOTAL TIME             3950   (TIMES ARE IN MILLISECONDS)
DATA STORAGE           6406      AVAILABLE CORE        6117      SYMBOL TABLE          6
7

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
      THIS  ROUTINE COMPUTES THE SYNDROMES S1,S2,S3,S4,S5.
      SUBROUTINE    SDROME(NR,NHT,NS,NS1,NS2,NS3,NS4,NS5,NCL,N,L,KL)
      DIMENSION    NR(N),NHT(N,L),NS(L),NS1(KL),NS2(KL),NS3(KL),NS4(KL),
     2NS5(KL)
      INTEGER    KNCT
      KNCT = 0
      DO  1  I = 1,L
      NS(I) = 0
      DO  1  J = 1,N
 1    NS(I) = NS(I)+NR(J)*NHT(J,I)
      DO  2  K = 1,L
 2    NS(K) = MODLL2(NS(K))
      DO  11  IJM = 1,L
      IF(NS(IJM).EQ.0)  KNOT = KNCT+1
 11   CONTINUE
      IF(KNOT.EQ.15)  NCL = 0
      IF(NCL.EQ.0)  RETURN
      DO  3  IX = 1,KL
      IP = IX+5
      IQ = IX+10
      NS1(IX) = NS(IX)
      NS3(IX) = NS(IP)
 3    NS5(IX) = NS(IQ)
      CALL    GFP(NS1,NS1,NS2,KL)
      CALL    GFP(NS2,NS2,NS4,KL)
      RETURN
      END
```

```
-------------------------------------------------------------------------+
    S I M U L T O R   F O R   B C H   D E C O D E R   U S I N G        C
             B E R L E K A M P # S    A L G O R I T H M                C
-------------------------------------------------------------------------C
  MAIN     PROGRAM
-------------------------------------------------------------------------C
  INTEGER     RV(31),HT(31,15),S(15),S1(5),S2(5),S3(5),S4(5),S5(5),
 1S6(5),SL0(5),SL1(5),SL2(5),SL3(5),TU0(5),TU1(5),TU2(5),TU3(5),
 2  ACC(5),RR(5),DIN(5),SIG0(5),SIG1(5),SIG2(5),SIG3(5),AL1(5),
 3AL2(5),AL3(5),ARC(5),SJ(6,5),DEL0(5),DEL1(5),DEL2(5),DEL3(5),ZE(5)
 4,ONE(5),R0(5),R1(5),R2(5),R3(5),R(5),CL
  DATA     AL1,AL2,AL3/0,1,5*0,1,5*0,1,0/
  DATA     ZE,ONE/5*0,1,4*0/
  CL = 4
  READ20,((HT(I,J),J=1,15),I=1,31)
  READ30,(RV(N),N=1,31)
  SYNDROME  COMPUTATION
  CALL     SDROME(RV,HT,S,S1,S2,S3,S4,S5,31,15,5)
  PRINT100,(S1(I),I=1,5)
  PRINT100,(S2(J),J=1,5)
  PRINT100,(S3(K),K=1,5)
  PRINT100,(S4(L),L=1,5)
  PRINT100,(S5(M),M=1,5)
  DO   110   IJ = 1,5
  SJ(1,IJ)=ONE(IJ)
  SJ(2,IJ),S1(IJ)
  SJ(3,IJ)=S2(IJ)
  SJ(4,IJ)=S3(IJ)
  SJ(5,IJ)=S4(IJ)
10  SJ(6,IJ)=S5(IJ)
```

```
C     COMPUTATION  OF  ERROR-LOCATOR-POLYNOMIAL  COEFFICIENTS  USING
C        BERLEKAMP#S  ALGORITHM.
      CALL    BKAMP(SJ,SL0,SL1,SL2,SL3,SIG0,SIG1,SIG2,SIG3,TU0,TU1,TU2,
     1TU3,DEL0,DEL1,DEL2,DEL3,ZF,ONE,ACC,RR,DIN,CL,6,5)
      PRINT100,(SIG3(I),I=1,5)
      PRINT100,(SIG2(J),J=1,5)
      PRINT100,(SIG1(K),K=1,5)
      PRINT100,(SIG0(L),L=1,5)
C     DECIDING  THE  ACTUAL  NO.OF  ERRORS  OCCURED  IN  THE  RECEIVED WORD.
      IF(CL.EQ.1)  PRINT111
      IF(CL.EQ.1)  PRINT222
      IF(CL.EQ.3)  PRINT333
      IF(CL.EQ.4)  PRINT444
      PRINT50,(RV(KL),KL=1,31)
C      CHIEN  SEARCH  AND  ERROR  CORRECTION.
      CALL    CHIEN(SIG0,SIG1,SIG2,SIG3,AL1,AL2,AL3,R0,R1,R2,R3,R,
     1ARC,RV,31,5)
      PRINT50,(RV(JK),JK=1,31)
C      DECODING  IS  COMPLETE  FOR  ONE  RECEIVED  WORD.
 20   FORMAT(5I1,5X,5I1,5X,5I1)
 30   FORMAT(31I1)
 50   FORMAT(1H0,20X,31I2)
 100  FORMAT(1H0,20X,5I2)
 111  FORMAT(1H0,20X,* THERE IS ONE ERROR IN THE TRANSMITTED WORD.*)
 222  FORMAT(1H0,20X,* THERE ARE TWO ERRORS IN THE TRANSMITTED WORD.*)
 333  FORMAT(1H0,20X,* THERE  ARE  THREE  ERRORS  IN  THE  TRANSMITTED
     1  WORD.*)
 444  FORMAT(1H0,20X,* THERE  ARE  MORE  THAN  THREE  ERRORS  IN  THE
     1 TRANSMITTED  WORD.*)
      STOP
      END
```

```
    BERLEKAMP#S ALGORITHM TO COMPUTE LOCATOR POLYNOMIAL COEFFICIENTS
    FROM  THE  SYNDROMES  S1,S2,S3,S4,S5,S6 COMPUTED BY#SDROME#ROUTINE
    SUBROUTINE    BKAMP(NS,NSL0,NSL1,NSL2,NSL3,NSG0,NSG1,NSG2,NSG3,
   ZNT0,NT1,NT2,NT3,ND0,ND1,ND2,ND3,NZ,NUN,MACC,NR,INDEL,NCL,NX,N)
    DIMENSION    NS(NX,N),NSL0(N),NSL1(N),NSL2(N),NSL3(N),NSG0(N),
   ZNSG1(N),NSG2(N),NSG3(N),NT0(N),NT1(N),NT2(N),NT3(N),ND0(N),ND1(N),
   ZND2(N),ND3(N),NZ(N),NUN(N),MACC(N),NR(N),INDEL(N)
    THIS ROUTINE COMPUTES THE ERROR-LOCATOR POLYNOMIAL
                   INITIAL  SOLUTION
    DO  1  I = 1,N
    NSL0(I) = NZ(I)
    NSL1(I) = NZ(I)
    NSG0(I) = NS(1,I)
    NSG1(I) = NZ(I)
    NSG2(I) = NZ(I)
    NSG3(I) = NZ(I)
    NT0(I) = NZ(I)
    NT1(I) = NS(1,I)
    NT2(I) = NZ(I)
 1  NT3(I) = NZ(I)
    MJ = 1
 2  DO  3  J = 1,N
    NSL3(J) = NSL1(J)
    NSL2(J) = NSL0(J)
    NSL1(J) = NS(MJ,J)
3   NSL0(J) = NS(MJ+1,J)
    IF(MJ.EQ.1)  GO  TO  14
    DO  13  LL = 1,N
    NT3(LL) = NT1(LL)
    NT2(LL) = NT0(LL)
    NT1(LL) = NZ(LL)
13  NT0(LL) = NZ(LL)
C                  COMPUTATION  OF  DELTA1 (2K )
14  CALL    GFP(NSL0,NSG0,ND0,N)
    CALL    GFP(NSL1,NSG1,ND1,N)
    CALL    GFP(NSL2,NSG2,ND2,N)
    CALL    GFP(NSL3,NSG3,ND3,N)
    DO  4  K = 1,N
 4  MACC(K) = MODUL2(ND0(K)+ND1(K)+ND2(K)+ND3(K))
C   COMPUTATION  OF  DELTA1(2K)  IS  COMPLETED.
    K1 = N
15  IF(MACC(K1).NE.0)  GO  TO  16
    K1 = K1-1
```

```
      IF(K1.NE.0)  GO  TO  15
      IF(MJ.EQ.1)  NCL = 1
      IF(MJ.EQ.3)  NCL = 2
      IF(MJ.EQ.5)  NCL = 3
      RETURN
      START  OF  OPERATION  SIGMA(2K+2)=SIGMA(2K)+DELTA1(2K)*7*TOU(2K)
16    CALL     GFP(MACC,NT0,NR,N)
      DO  5  L = 1,N
5     NSG0(L) = MODUL2(NSG0(L)+NR(L))
      CALL     GFP(MACC,NT1,NR,N)
      DO  6  M = 1,N
6     NSG1(M) = MODUL2(NSG1(M)+NR(M))
      CALL     GFP(MACC,NT2,NR,N)
      DO  7  IJ = 1,N
7     NSG2(IJ) = MODUL2(NSG2(IJ)+NR(IJ))
      CALL     GFP(MACC,NT3,NR,N)
      DO  8  IK = 1,N
8     NSG3(IK) = MODUL2(NSG3(IK)+NR(IK))
      OPERATION  SIGMA(2K+2) = SIGMA(2K)+DELTA1*Z*TOU(2K)  IS OVER
      COMPUTATION  OF  DELTA1 INVERSE  AND  THERE  BY  TOU(2K+2)
      CALL     INVERS(NUN,MACC,INDEL)
      CALL     GFP(NSG0,INDEL,NR,N)
      DO  9  IL = 1,N
9     NT0(IL) = MODUL2(NT0(IL)+NR(IL))
      CALL     GFP(NSG1,INDEL,NR,N)
      DO  10  IM = 1,N
10    NT1(IM) = MODUL2(NT1(IM)+NR(IM))
      CALL     GFP(NSG2,INDEL,NR,N)
      DO  11  IN = 1,N
11    NT2(IN) = MODUL2(NT2(IN)+NR(IN))
      CALL    GFP(NSG3,INDEL,NR,N)
      DO  12  IO = 1,N
12    NT3(IO) = MODUL2(NT3(IO)+NR(IO))
      COMPUTATION  OF  TOU(2K+2) =(DELTA1)-1  *SIGMA(2K) * 7 IS  OVER
      SHIFT THE S-COLUMN UPWARDS TWICE AND TOU-COLUMN TWICE  UPWARDS
                     PROCEED  WITH  THE  NEXT  ITERATION  OF  ALGORITHM
      MJ = MJ+2
      IF(MJ.LE.5)  GO  TO  2
      RETURN
      END
```

```
C     S I M U L T O R   P R O G R A M   FOR 'V D H - B' A L G O R I T H M      C
C     MAIN  PROGRAM
C-----------------------------------------------------------------------------C
C     A0,A1,A2,A3,  ARE  ELEMENTS  OF  GF(2**5)
C     S1,S3,S5  ARE  SYNDROMES   AND   T1,T3,T5  ARE  TRANSFORMED  SYNDROMES
C     RV  --  RECEIVED  WORD.
C     HT(31,15)  ---  TRANSPOSE  OF  PARITY  CHECK  MATRIX  OF  (31,16)  CODE.
C     SIG0,SIG1,SIG2,SIG3  ARE  COEFFICIENTS  OF  THE  ERROR-LOCATOR-POLYNOMIAL
C     NERR  ---   ACTUAL  NUMBER  OF  ERRORS
C-----------------------------------------------------------------------------C
      INTEGER    RX(31)
      INTEGER    A0(5),A1(5),A2(5),A3(5),S(15),S1(5),S3(5),S5(5),T3(5),
     1T5(5),SIG0(5),SIG1(5),SIG2(5),SIG3(5),R0(5),R1(5),R2(5),R3(5),R(5)
     2,ACC(5),RV(31),HT(31,15),NERR
      DATA   A0,A1,A2,A3,SIG0    /1,5*0,1,5*0,1,5*0,1,0,1,4*0/
      PRINT50
      READ10,((HT(I,J),J=1,15),I=1,31)
      DO  9  IJK = 1,1000
      READ20,(RV(K),K=1,31)
      DO  141 IPK = 1,31
 141  RX(IPK) = RV(IPK)
C     COMPUTATION  OF  SYNDROMES .  IF  SYNDROMES  VANISH  THERE  ARE  NO
C     ERRORS  IN  THE  TRANSMITTED  WORD.  PROCEED  TO  THE  NEXT  RECEIVED
C     WORD.
      CALL    SDROME(RV,HT,S,S1,S3,S5,NERR,31,15,5)
      IF(NERR.EQ.0)  GO  TO  8
C     STEP1  OF  VDH-B  ALGORITHM
      CALL    STEP1(S1,S3,S5,T3,T5,SIG1,SIG2,SIG3,NERR,5)
      IF(NERR.EQ.1)  GO  TO  7
C     STEP2  OF  VDH-B  ALGORITHM
```

```
      CALL     STEP2(S1,S3,T3,T5,A0,SIG1,SIG2,SIG3,NERR,5)
      IF(NERR.EQ.2)  GC  TC  7
      IF(NERR.EQ.3)  GC  TC  6
      IF(NERR.EQ.4)  PRINT160
      GC  TC  9
C     STEP3  CF  VCH-B  ALGCRITHM
    6 CALL     STEP3(A0,A1,A3,S1,S3,T3,T5,SIG1,SIG2,SIG3,NERR,5)
      IF(NERR.EQ.3)  GC  TC  7
      PRINT80
      GC  TC  9
C     CHIEN  SEARCH
    7 CALL     CHIEN(SIG0,SIG1,SIG2,SIG3,A1,A2,A3,R0,R1,R2,R3,R,ACC,RV,
     1NERR,31,5)
      PRINT150,(RX(IXL),IXL=1,31),(RV(JXL),JXL=1,31)
    8 IF(NERR.EQ.0)  PRINT40
    9 CCNTINUE
   10 FCRMAT(5I1,5X,5I1,5X,5I1)
   20 FCRMAT(11X,31I1)
   40 FCRMAT(1H0,20X,* NC ERRCRS IN THE TRANSMITTED WORD *)
   50 FCRMAT(1H0,40X,* RECEIVED WORD *,20X,* CORRECTED WCRD *)
   80 FCRMAT(1H0,20X,* NC.CF ERRORS ARE UNCORRECTABLE *)
  150 FCRMAT(1H0,32X,31I1,8X,31I1)
  160 FCRMAT(1H0,20X,* NUMBER  OF  ERRORS  ARE  MORE  THAN  THREE  IN
     1 THE  TRANSMITTEC  WCRC *)
      STCP
      END
```

```
2     TCTAL TIME          13200   (TIMES ARE IN MILLISECONDS)
1     CATA STCRAGE         6407     AVAILABLE CCRE        6069     SYMBOL TABLE
 101177
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
      SUBROUTINE    STEP1(NS1,NS3,NS5,NT3,NT5,NSG1,NSG2,NSG3,NOER,N)
      DIMENSION     NS1(N),NS3(N),NS5(N),NT3(N),NT5(N),NSG1(N),NSG2(N),
     1NSG3(N)
      INTEGER    A(5),B(5),C(5)
      NOER = 0
      COMPUTING  TRANSFORMED  SYNDROMES
      CALL    GFP(NS1,NS1,A,N)
      CALL    GFP(NS1,A,B,N)
      CALL    GFP(A,B,C,N)
      DO  1  I = 1,N
      NT3(I) = MOD2(NS3(I)+B(I))
    1 NT5(I) = MOD2(NS5(I)+C(I))
      CHECKING  FOR  THE  OCCURANCE  OF  A  SINGLE  ERROR
      DO  2  J = 1,N
      IF(NT3(J).NE.0)  GO  TO  4
      IF(NT5(J).NE.0)  GO  TO  4
    2 CONTINUE
      NOER = 1
      DO  3  K = 1,N
      NSG1(K) = NS1(K)
      NSG2(K) = 0
    3 NSG3(K) = 0
C      PROCEED  TO  THE  EXECUTION  OF  STEP2.
    4 RETURN
      END
```

```
C     STEP2  CF  VCH-B  ALGCRITHM
      SUBRCUTINE    STEP2(NS1,NS3,NT3,NT5,NAO,NSG1,NSG2,NSG3,KOUNT,N)
      DIMENSICN    NS1(N),NS3(N),NT3(N),NT5(N),NAO(N),NSG1(N),NSG2(N),
     1NSG3(N)
      INTEGER    A(5),B(5),C(5),D(5),E(5),F(5),FD(5),P(5),Q(5),R(5),S(5)
     1,T(5),TT(5),TVT(5),PN(5),QN(5),PNR,TRP
      INTEGER    MX(5),NX(5)
      KOUNT = 0
      DC  7  LX = 1,N
      MX(LX) = NS1(LX)
    7 NX(LX) = NT3(LX)
      CALL    INVERS(NAO,NX,A)
      CALL    GFP(A,A,B,N)
      CALL    GFP(B,B,C,N)
      CALL    GFP(A,C,D,N)
      CALL    GFP(NT5,NT5,E,N)
      CALL    GFP(NT5,E,F,N)
      CALL    GFP(F,D,FD,N)
      DC  1  I = 1,N
    1 P(I) = MOD2(NAO(I)+FD(I))
C     CCMPUTATICN  CF  TR(T5**3/T3**5)+1)
      CALL    TRACE(P,TRP,N)
C     IF  THIS  TRACE  EQUALS  '1'  FOUR  CR  MORE  ERRORS  HAVE  CCCURED.
      IF(TRP.EQ.1)  GC  TC  6
C     CCMPUTING  CF  SIGMA'(S1)   AND  TR(T3/S1**3)
C     IF  BCTH   SIGMA'(S1)  AND  TR(T3/S1**3)  VANISH  NC.CF.ERRORS = 2
      CALL    GFP(NT5,A,G,N)
      CALL    GFP(G,NS1,R,N)
      DC  2  J = 1,N
    2 S(J) = MOD2(R(J)+NS3(J))
```

```
      DO  3  K = 1,N
      IF(S(K).NE.0)  GO  TO  5
   3  CONTINUE
      CALL     INVERS(NAO,MX,T)
      CALL     GFP(T,T,TT,N)
      CALL     GFP(T,TT,TVT,N)
      CALL     GFP(NT3,TVT,PN,N)
      CALL     TRACE(PN,PNR,N)
      IF(PNR.EQ.1)  GO  TO  5
      KOUNT = 2
      CALL     GFP(NT3,T,QN,N)
      DO  4  L = 1,N
      NSG1(L) = NS1(L)
      NSG2(L) = QN(L)
   4  NSG3(L) = 0
      RETURN
C     NUMBER  OF  ERRORS  IN  THE  RECEIVED  WORD  EQUALS  3
   5  KOUNT = 3
      RETURN
   6  KOUNT = 4
      RETURN
      END
   TOTAL TIME          12283   (TIMES ARE IN MILLISECONDS)
   DATA STORAGE         6407     AVAILABLE CORE       6069     SYMBOL TABLE
01177

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
C     STEP3  OF  VDH-B  ALGORITHM
      SUBROUTINE    STEP3(NA0,NA1,NA3,NS1,NS3,NT3,NT5,NSG1,NSG2,NSG3,
     1KOUNT,N)
      DIMENSION    NA0(N),NA1(N),NA3(N),NS1(N),NS3(N),NT3(N),NT5(N),
     1NSG1(N),NSG2(N),NSG3(N)
      INTEGER    A(5),B(5),C(5),D(5),E(5)
      KOUNT = 0
      DO  3  IP = 1,N
    3 E(IP) = NT3(IP)
      CALL    INVERS(NA0,E,A)
      CALL    GFP(NT5,A,B,N)
C     SEARCH  FOR  THE  ROOTS  OF  SIG'(X) = X**3+(T5/T3)+T3  IN  GF(2**5)
C     IS  DONE  BY  THE 'ROOTS' ROUTINE
      CALL    ROOTS(NA0,NA1,NA3,NT3,B,NOR,N)
C   , IF  THE  SEARCH  SUCCEEDS  NOR = 1 . HENCE  IF  NOR = 1  THEN
C     SIG(X) = SIG'(X+S1).   STOP.   ELSE GIVE A MESSAGE THAT NO. OF ERRORS
C     ARE MORE THAN 3 AND HENCE UNCORRECTABLE.
      IF(NOR.EQ.1)  GO  TO  1
      KOUNT = KOUNT+4
      RETURN
    1 KOUNT = KOUNT+3
      CALL    GFP(NS1,NS1,C,N)
      CALL    GFP(NS1,B,D,N)
      DO  2  I = 1,N
      NSG1(I) = NS1(I)
      NSG2(I) = MOD2(B(I)+C(I))
    2 NSG3(I) = MOD2(NS3(I)+D(I))
      RETURN
      END
```

```
C     THIS ROUTINE SEARCHES FOR THE ROOTS OF A THIRD DEGREE
C     POLYNOMIAL OVER GF(2**5) .
      SUBROUTINE ROOTS(KA0,KA1,KA3,KT3,KB,KOR,K)
C     KA0,KA1,KA3 ARE THE NONZERO ELEMENTS OF GF(2**5) AND KA0 IS
C     IS THE UNIT ELEMENT OF GF(2**5)
      DIMENSION KA0(K),KA1(K),KA3(K),KT3(K),KB(K)
      INTEGER A(5),B(5),C(5),D(5),E(5)
      KOR = 0
      NROOT = 0
      I = 1
      DO 1 J = 1,K
      A(J) = KA0(J)
    1 B(J) = KB(J)
    2 DO 3 L = 1,K
    3 C(L) = MOD2(A(L)+B(L)+KT3(L))
      DO 4 M = 1,K
      IF(C(M).NE.0) GO TO 5
    4 CONTINUE
      NROOT = NROOT+1
C     IF THE NUMBER OF ROOTS OF SIG'(X) IN GF(2**5) ARE THREE
C     THEN NROOT = 3, AND SIG(X) = SIG'(X+S1) . ELSE THERE ARE MORE
C     THAN THREE ERRORS IN THE RECEIVED WORD.
      IF(NROOT.EQ.3) GO TO 7
    5 I = I+1
      CALL GFP(A,KA3,D,K)
      CALL GFP(B,KA1,E,K)
      DO 6 NK = 1,K
      A(NK) = D(NK)
    6 B(NK) = E(NK)
      IF(I.LE.31) GO TO 2
      RETURN
    7 KOR = KOR+1
      RETURN
```

```
C     THIS ROUTINE COMPUTES THE GF(2) TRACE OF AN ELEMENT OF GF(2**5)
C     ALL ELEMENTS OF GF(2**5) ARE REPRESENTED AS BINARY 5-TUPLES.
      SUBROUTINE    TRACE(NP,NPR,N)
      DIMENSION    NP(N)
      INTEGER    A(5),B(5),C(5),D(5),E(5)
      NPR = 0
      CALL    GFP(NP,NP,A,N)
      CALL    GFP(A,A,B,N)
      CALL    GFP(B,B,C,N)
      CALL    GFP(C,C,D,N)
      DO 1  I = 1,N
    1 E(I) = MOD2(NP(I)+A(I)+B(I)+C(I)+D(I))
      IF(E(1).EQ.1)  NPR = 1
      RETURN
      END
C      THIS SUBPROGRAM REDUCES THE NUMBER 'NN' TO NN MOD2 .
      FUNCTION    MOD2(NN)
      MOD2 = IABS(NN-(NN/2)*2)
      RETURN
      END
   TOTAL TIME            4916   (TIMES ARE IN MILLISECONDS)
   DATA STORAGE          6406     AVAILABLE CORE        6117     SYMBOL TABLE
111077
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXX
```

```
C     THIS  ROUTINE  DOES  CHIEN  SEARCH  ON  THE  ERROR-LOCATOR-POLYNIMIAL
C     NG0,NG1,NG2,NG3  ARE  THE  COEFFICIENTS  OF  THE  ERROR-LOCATOR-POLY
C     NOMIAL. NA1,NA2,NA3  ARE  FIELD  ELEMENTS .
C     NRV---  THE  RECEIVED  VECTOR
      SUBROUTINE    CHIEN(NG0,NG1,NG2,NG3,NA1,NA2,NA3,NR0,NR1,NR2,
     1NR3,NX,NAP,NRV,KORT,M,N)
      DIMENSION    NG0(N),NG1(N),NG2(N),NG3(N),NA1(N),NA2(N),NA3(N),
     1NR0(N),NR1(N),NR2(N),NR3(N),NX(N),NAP(N),NRV(M)
      MOOD = 0
C     INUTIALLY  NG1,...,NG3  ARE  PLACED  IN  4  REGISTERS  . THEN FOR
C     EACH  CLOCK CYCLE THE CONTENTS OF THESE REGISTERS ARE MULTIPLIED
C     BY  NA0,NA1,NA2,NA3  RESPECTIVELY  AND  ADDED  BIT-WISE  MOD2.
C     WHEN  THIS  SUM  IS  ZERO  THE  CORRESPONDING  'BIT'  COMING  OUT
C     BUFFER  IS  IN  ERROR  AND  IS  COMPLEMENTED, THUS  CORRECTING  THE
C     ERROR.
      DO  1  I = 1,N
      NR0(I) = NG0(I)
      NR1(I) = NG1(I)
      NR2(I) = NG2(I)
    1 NR3(I) = NG3(I)
      L = M
    2 CALL    GFP(NR1,NA1,NX,N)
      DO  4  IP = 1,N
    4 NR1(IP) = NX(IP)
      CALL    GFP(NR2,NA2,NX,N)
      DO  5  JP = 1,N
    5 NR2(JP) = NX(JP)
      CALL    GFP(NR3,NA3,NX,N)
      DO  6  KP = 1,N
    6 NR3(KP) = NX(KP)
```

```
 7  DO  8  LP = 1,N
 8  NAP(LP) = MOD2(NRO(LP)+NR1(LP)+NR2(LP)+NR3(LP))
    K1 = N
 9  IF(NAP(K1).NE.0)  GO  TO  10
    K1 = K1-1
    IF(K1.NE.0)  GO  TO  9
C    ERROR  CORRECTION.
    NRV(L) = MOD2(NRV(L)+1)
    MOOD = MOOD+1
    IF(MOOD.EQ.KONT)  GO  TO  11
10  L = L-1
    IF(L.GE.1)  GO  TO  2
11  RETURN
    END
  TOTAL TIME           10466   (TIMES ARE IN MILLISECONDS)
  DATA STORAGE          6407     AVAILABLE CORE       6073     SYMBOL TABLE
111077

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
C     THIS ROUTINE CHECKS WHETHER OR NOT THE DECODER OUTPUT IS A CODE WORD,
      SUBROUTINE    CHECK(MCV,MHT,MS,MANJU,IL,JL)
      DIMENSION    MCV(IL),MHT(IL,JL),MS(JL)
      MANJU = 0
      NBINDU = 0
      DO  1  I = 1,JL
      MS(I) = 0
      DO  1  J = 1,IL
    1 MS(I) = MS(I) + MCV(J)*MHT(J,I)
      DO  2  K = 1,JL
    2 MS(K) = MODUL2(MS(K))
      DO  3  LL = 1,JL
      IF(MS(LL).EQ.0)  NBINDU = NBINDU+1
    3 CONTINUE
      IF(NBINDU.EQ.15)  MANJU = 1
      RETURN
      END
   TOTAL TIME            4483   (TIMES ARE IN MILLISECONDS)
   DATA STORAGE          6406     AVAILABLE CORE        6117      SYMBOL TABLE
101277

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
C     THIS ROUTINE COMPUTES THE GF PRODUCT OF TWO ELEMENTS FOR GF(2**5)
      SUBROUTINE    GFP(NA,NB,NAC,NL)
      DIMENSION    NA(NL),NB(NL),NAC(NL)
      INTEGER    R(5)
      DO  1  I = 1,NL
    1 NAC(I) = 0
      IJ = NL
    2 IF(NA(IJ).NE.1)  GO  TO  4
      DO  3  J = I,NL
    3 NAC(J) = MODUL2(NAC(J)+NB(J))
    4 IF(IJ.EQ.1)  GO  TO  6
      R(1) = NAC(NL)
      R(2) = NAC(NL-4)
      R(3) = MODUL2(NAC(NL-3)+NAC(NL))
      R(4) = NAC(NL-2)
      R(5) = NAC(NL-1)
      DO  5  K = 1,NL
    5 NAC(K) = R(K)
      IJ = IJ-1
      IF(IJ.NE.0)  GO  TO  2
    6 RETURN
      END
```

TOTAL TIME   5300   (TIMES ARE IN MILLISECONDS)
DATA STORAGE   6406   AVAILABLE CORE   6117   SYMBOL TABLE
101177

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

```
C     THIS ROUTINE COMPUTES THE GF INVERSE OF AN ELEMENT OF GF(2**5)
      SUBROUTINE   INVERS(NON,NP,INV)
      DIMENSION   NON(5),NP(5),INV(5)
      INTEGER   R(5),S(5)
      DO 10  I = 1,5
 10   INV(I) = NON(I)
 20   R(1) = NP(5)
      R(2) = NP(1)
      R(3) = MOD2(NP(2)+NP(5))
      R(4) = NP(3)
      R(5) = NP(4)
      DO 30  J = 1,5
 30   NP(J) = R(J)
      S(1) = INV(5)
      S(2) = INV(1)
      S(3) = MOD2(INV(2)+INV(5))
      S(4) = INV(3)
      S(5) = INV(4)
      DO 40  K = 1,5
 40   INV(K) = S(K)
      IJ = 5
 50   IF(NP(IJ).NE.NON(IJ))   GO TO 20
      IJ = IJ-1
      IF(IJ.NE.0)  GO  TO  50
      RETURN
      END
      FUNCTION   MOD2(KK)
      MOD2 = IABS(KK-(KK/2)*2)
      RETURN
      END
```

```
0     TOTAL TIME          7133   (TIMES ARE IN MILLISECONDS)
6     DATA STORAGE        6406     AVAILABLE CORE       6113      SYMBOL TABLE
101177
```

Date Slip

This book is to be returned on the date last stamped.